COMPACTLY SUPPORTED BASIS FUNCTIONS AS SUPPORT VECTOR KERNELS: CAPTURING FEATURE INTERDEPENDENCE IN THE EMBEDDING SPACE

PETER WITTEK

(M.Sc. Mathematics, M.Sc. Engineering and Management)

A THESIS SUBMITTED FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

NATIONAL UNIVERSITY OF SINGAPORE

2010

Acknowledgments

I am thankful to Professor Tan Chew Lim, my adviser, for giving all the freedom I needed in my work, and despite his busy schedule he was always ready to point out the mistakes I made and to offer his help to correct them.

I am also grateful to Professor Sándor Darányi, my long-term research collaborator, for the precious time he has spent on working with me. Many fruitful discussions with him have helped to improve the quality of this thesis.

Contents

Summary

Dependencies between variables in a feature space are often considered to have a negative impact on the overall effectiveness of a machine learning algorithm. Numerous methods have been developed to choose the most important features based on the statistical properties of features (feature selection) or based on the effectiveness of the learning algorithm (feature wrappers). Feature extraction, on the other hand, aims to create a new, smaller set of features by using relationship between variables in the original set. In any of these approaches, reducing the number of features may also increase the speed of the learning process, however, kernel methods are able to deal with very high number of features efficiently. This thesis proposes a kernel method which keeps all the features and uses the relationship between them to improve effectiveness.

The broader framework is defined by wavelet kernels. Wavelet kernels have been introduced for both support vector regression and classification. Most of these wavelet kernels do not use the inner product of the embedding space, but use wavelets in a similar fashion to radial basis function kernels. Wavelet analysis is typically carried out on data with a temporal or spatial relation between consecutive data points.

The new kernel requires the feature set to be ordered, such that consecutive features are related either statistically or based on some external knowledge source; this relation is meant to act in a similar way as the temporal or spatial relation on

other domains. The thesis proposes an algorithm which performs this ordering.

The ordered feature set enables to interpret the vector representation of an object as a series of equally spaced observations of a hypothetical continuous signal. The new kernel maps the vector representation of objects to the L_2 function space, where appropriately chosen compactly supported basis functions utilize the relation between features when calculating the similarity between two objects.

Experiments on general-domain data sets show that the proposed kernel is able to outperform baseline kernels with statistical significance if there are many relevant features, and these features are strongly or loosely correlated. This is the typical case for textual data sets.

The suggested approach is not entirely new to text representation. In order to be efficient, the mathematical objects of a formal model, like vectors, have to reasonably approximate language-related phenomena such as word meaning inherent in index terms. On the other hand, the classical model of text representation, when it comes to the representation of word meaning, is approximate only. Adding expansion terms to the vector representation can also improve effectiveness. The choice of expansion terms is either based on distributional similarity or on some lexical resource that establishes relationships between terms. Existing methods regard all expansion terms equally important. The proposed kernel, however, discounts less important expansion terms according to a semantic similarity distance. This approach improves effectiveness in both text classification and information retrieval.

List of Figures

List of Tables

List of Symbols

Symbol	Definition
b	Length of the support of a basis function
$b(t)$	A basis function of L_2
C	Set of classes
c_k	The kth class
$d(.,.)$	A distance function
f_i	A feature
ϕ	A kernel mapping
Φ	A classifier function
K	A kernel matrix
$K(.,.)$	A kernel function
M	The number of features
N	The number of training objects in a collection
$N(c)$	The number of training objects in category c
$N(x)$	The number of features in object x
N_i	The number of objects containing feature f_i at least once
S	Semantic matrix
s_i	A sense of a term
t	General dummy variable, $t \in \mathbb{R}$

Symbol	Definition
$\text{sen}(t)$	The set of senses of a term t
$\mathbf{w}$	Normal vector of a separating hyperplane
X	A finite dimension real-valued vector space
x_i	An object subject to classification
$\mathbf{x}_i$	A finite dimensional vector representation of x_i
x_{ij}	One element of X

List of Publications Related to the Thesis

Wittek, P., S. Daranyi, M. Dobreva. 2010. Matching Evolving Hilbert Spaces and Language for Semantic Access to Digital Libraries *Proceedings of Digital Libraries for International Development Workshop. In conjuction with JCDL-10, 12th Joint Conference on Digital Libraries.*

Daránvi, S., P. Wittek, M. Dobreva. 2010. Toward a 5M Model of Digital Libraries. *Proceedings of ICADL-10, 12th International Conference on Asia-Pacific Digital Libraries.*

Wittek, P., C.L. Tan. 2009. A Kernel-based Feature Weighting for Text Classification. *Proceedings of IJCNN-09, 22nd International Joint Conference on Neural Networks.* Atlanta, GA, USA, June.

Wittek, P., S. Daránvi, C.L. Tan. 2009. Improving Text Classification by a Sense Spectrum Approach to Term Expansion. *Proceedings of CoNLL-09, 13th Conference on Computational Natural Language Learning.* Boulder, CO, USA, June.

Wittek, P., C.L. Tan, S. Daránvi. 2009. An Ordering of Terms Based on Semantic Relatedness. In H. Bunt, editor, *Proceedings of IWCS-09, 8th International Conference on Computational Semantics.* Tilburg, The Netherlands, January. Springer

Wittek, P. and S. Darányi. 2007. Representing word semantics for IR by continuous functions. In S. Dominich and F. Kiss, editors, *Studies in Theory of Information Retrieval. Proceedings of ICTIR-07, 1st International Conference of the Theory of Information Retrieval*, pages 149–155, Budapest, Hungary, October. Foundation for Information Society.

Chapter 1

Introduction

Machine learning has been central to artificial intelligence from the beginning, and one of the most fundamental questions of this field is how to represent objects of the real world so that mathematical algorithms can be deployed on them for processing. Feature generation, selection, weighting, expansion, and ultimately, feature engineering have a vast literature addressing both the general case and feature sets with certain characteristics.

The lack of capacity of current machine learning techniques to handle feature interactions have fostered significant research effort. Techniques were developed to enhance the power of data representation used in learning, however, most existing techniques focus on feature interactions between nominal or binary features (Donoho and Rendell, 1996). This thesis focuses on continuous features identifying a gap between feature weighting and feature expansion, and attempts to fill it within the framework of kernel methods. The proposed kernel is particularly efficient in incorporating prior knowledge in the classification without additional storage needs. This characteristic of the kernel is useful on emerging computing platforms such as cloud environment or general purpose programming on graphics processing units. This section puts the present work into perspective.

1.1 Supervised Machine Learning for Classification

Supervised machine learning is a technique for learning a linear or nonlinear function from training data consisting of pairs of input objects and matching outputs. If the outputs are discrete labels, classes, the learning task is called supervised classification or categorization. The objective of the learner is to predict the value of the function for any valid input object after having seen a number of training examples, that is, the learner has to generalize from the presented data to unseen situations.

Generally, building an automated classification system consists of two key subtasks. The first task is representation: converting the properties of input objects to a compact form so that they can be further processed by the learning algorithm. Another task is to learn the model itself which is then applied to classify unlabeled input objects.

1.2 Feature Selection and Weighting

Determining the input feature representation is essential, since the accuracy of the learned function depends strongly on how the input object is represented. Typically, the input object is transformed into a feature vector, which contains a number of features that are descriptive of the object. Features are the individual measurable heuristic properties of the phenomena being observed. The features are not necessarily independent. For instance, in text categorization, the features are terms of the document collection (the input objects), with a range of different types of dependencies between the terms: synonymy, antnonymy, and other semantic relations (Manning and Schütze, 1999).

Certain measurable features are not necessarily relevant for a given classification task. For instance, given the task of distinguishing cancer versus normal patterns from mass-spectrometric data, the number of features is very large with only a fraction of the features being relevant (Guyon et al., 2005). Choosing too many features that are not independent or not relevant might lead to overfitting the training data. To address these two problems, feature selection and feature extraction methods have been developed to reduce the number of dimensions (Guyon, Elisseefi, and Kaelbling, 2003).

Feature selection algorithms apply some criteria to eliminate features and produce a reduced set thereof. Feature weighting algorithms are more sophisticated: instead of assigning one or zero to a feature (that is, keeping the feature or eliminating it), continuous weights are calculated for each feature. However, once these weights are calculated, they remain rigid during the classification process.

1.3 Feature Expansion

In many cases, feature enrichment is used, as opposed to feature selection. This approach can help when the feature space is sparse. For instance, in text categorization, vectors are sparse, with one to five per cent of the entries being nonzero (Berry, Dumais, and O'Brien, 1995). When an unseen document is to be classified, term expansion can be used to improve efficiency: terms that are related to terms of the document are added to the vector representation (Rodriguez and Hidalgo, 1997; Ureña López, Buenaga, and Gómez, 2001). This method is not as rigid as feature weighting, since it dynamically adds new features to the representation, but it treats all expansion features as if they were equally important or unimportant. The latter consideration resembles feature selection, which treats a feature as either important or absolutely irrelevant. While it is possible to introduce weighting schemes into feature expansion, these methods tend to be heuristic or domain

dependent, hence the need for a more generic approach.

1.4 Motivation for a New Kernel

This thesis offers a representation that enriches the original feature set in a similar vein to term expansion, but weights the expansion features individually.

Wavelet kernels have been introduced for both support vector regression and classification. These wavelet kernels use wavelets in a similar fashion to radial basis function kernels, and they do not use the inner product of the embedding space. Wavelet analysis is typically carried out on data with a temporal or spatial relation between consecutive features; since general data sets do not necessarily have these relations between features, the deployment of wavelet analysis tools have been limited in many fields.

This thesis argues that it is possible to order the features of a general data set so that consecutive features are statistically or otherwise related to each other, thus interpreting the vector representation of an object as a series of equally spaced observations of a hypothetical continuous signal. By approximating the signal with compactly supported basis functions (CSBF) and employing the inner product of the embedding L_2 space, we gain a new family of wavelet kernels.

Once the representation is created, a learning algorithm learns the function from the training data. Kernel methods and support vector machines have emerged as universal learners having been applied to a wide range of linear and nonlinear classification tasks (Cristianini and Shawe-Taylor, 2000). The proposed representation is proved to be a valid kernel for support vector machines, hence it can be applied in the same wide range of scenarios.

1.5 Structure of This Thesis

This thesis is organized as follows. Chapter 2 reviews the relevant literature by first looking at feature selection and feature extraction, then proceeding to the most common supervised machine learning algorithms for classification. The literature review establishes the broader context of the research presented in the rest of the thesis.

Chapter 3 introduces the proposed CSBF kernels. Compactly and non-compactly supported wavelet kernels have already been developed, the chapter uses this context to derive a new family of wavelet kernels. These kernels requires the feature set to be ordered, such that consecutive features are related either statistically or based on some external knowledge source. This chapter suggests an algorithm which performs the ordering. Once the order is generated, the new kernel maps the vector representation of objects to the L_2 function space, where appropriately chosen compactly supported basis functions utilize the relation between features when calculating the similarity between two objects. The choice of basis functions is essential, it is argued that nonorthogonal basis functions serve the purpose better. The chapter discusses the mathematical validity of the new kernels, as well as their computational complexity, issues with efficient implementation, and presents experimental results. The results show that the proposed kernels may outperform baseline methods if the number of relevant features is high, and the features are also highly correlated.

A special field of supervised classification typically has such feature sets: text categorization. Two terms can be related in many ways: they can be synonyms, one can be in an instance-of relation with the other, they could be syntactically related, etc. Chapter 4 offers insights into term relations, term similarity, and applies the proposed kernels in the domain of text classification, showing significant improvements over baseline methods.

Finally, Chapter 5 outlines the key contributions once again, and concludes the thesis.

Chapter 2

Literature Review

This chapter reviews the relevant literature to lay down the theoretical foundations of the present work and define the broader context of the rest of chapters.

Feature selection and feature extraction are fundamental methods in machine learning (Section 2.1), reducing complexity and often improving efficiency. Feature weighting is a subclass of feature selection algorithms (Section 2.1.1.2). It does not reduce the actual dimension, but weights features according to their importance. However, the weights are rigid, they remain constant for every single input instance.

Machine learning has a vast literature (Section 2.2). In the past decade, support vector machines and kernel methods emerged as compelling algorithms in most domains, due to their ability to work with extremely high dimensional spaces, their scalability, and their robustness (Section 2.2.6). Kernel methods are able to map finite dimensional spaces to infinite dimensional spaces, a quality that is used by very few kernel types.

2.1 Feature Selection and Feature Extraction

Several factors affect the success of machine learning on a given task. The representation and quality of the example data are first and foremost. Having more features should result in more discriminating power and thus higher effectiveness in machine learning. However, practical experience with machine learning algorithms has shown that this is not always the case. Many learning algorithms can be viewed as making an estimate of the probability of the class label given a set of features. The distribution is complex and high dimensional. Normally, induction is often performed based on limited training data, thus making estimating the probabilistic parameters difficult. To avoid overfitting the training data, many algorithms employ the so-called Occam's Razor (Gamberger and Lavrac, 1997) bias to build a simple model that still achieves some acceptable level of performance on the training data. This bias often leads certain algorithms to prefer a small number of predictive features over a large number of features that, if used in the proper combination, are as much or even more predictive than the full set of features. If irrelevant and redundant information is present or the data is noisy or unreliable, then learning during the training phase is more difficult.

Feature selection and feature extraction have become the focus of much research in areas of application for which data sets with tens or hundreds of thousands of features are available. These areas among others include text processing of Internet documents, gene expression array analysis, and combinatorial chemistry. Feature subset selection is the process of identifying and removing as much irrelevant and redundant information as possible. Feature extraction, on the other hand, creates a new, reduced set of features which combines elements of the original feature set. The objective of feature selection and feature extraction is three-fold: improving the prediction performance of the predictors, providing algorithmically faster predictors, and providing a better understanding of the underlying process

that generated the data (Guyon, Elisseefi, and Kaelbling, 2003).

Recent research has shown many common machine learning algorithms to be adversely affected by irrelevant or redundant training information. The simple nearest neighbor algorithm is sensitive to irrelevant features – its sample complexity (number of training examples needed to reach a given accuracy level) grows exponentially with the number of irrelevant features (Langley and Sage, 1994b; Langley and Sage, 1997; Aha, Kibler, and Albert, 1991). Sample complexity for decision tree algorithms can grow exponentially on some concepts as well. The naïve Bayes classifier can be adversely affected by redundant features due to its assumption that features are independent given the class (Langley and Sage, 1994a). Decision tree algorithms such as C4.5 can sometimes overfit training data, resulting in large trees. In many cases, removing irrelevant and redundant information can result in C4.5 producing smaller trees (Kohavi and John, 1997). However, in the case of support vector machines, feature selection seems to have less impact on the efficiency (Weston et al., 2000).

The potential benefits of feature selection and feature extraction include: facilitating data visualization and data understanding, reducing the measurement and storage requirements, reducing training and utilization times, defying the curse of dimensionality to improve prediction performance (Guyon, Elisseefi, and Kaelbling, 2003). Methods differ in which aspect they put more emphasis on. The problem of finding or ranking all potentially relevant features is slightly different. Selecting the most relevant features tends to be suboptimal for building a predictor, especially if the features are redundant. Similarly, a subset of highly predictive features may exclude many redundant, but relevant, features (Guyon, Elisseefi, and Kaelbling, 2003).

2.1.1 Feature Selection Algorithms

Feature selection has long been a research area within statistics and pattern recognition (Devijver and Kittler, 1982; Miller, 1990). Machine learning has taken inspiration and borrowed from both pattern recognition and statistics. Feature selection algorithms (with a few notable exceptions) perform a search through the space of features or feature subsets, and thus must address four basic issues affecting the nature of the search (Langley, 1994):

1. Selection of the starting point. Selecting a point in the feature subset space from which to begin the search can affect the direction of the search. One option is to begin with no features and successively add features; the search proceeds forward through the search space. Conversely, the search can also begin with all features and successively remove some of them; the search proceeds backward through the search space. A third alternative is to begin somewhere in the middle and move outwards from this point.

2. Search organization. An exhaustive, brute-force search of the feature subspace is not desirable for all but a small initial number of features. With M initial features, there are 2^M possible subsets. Heuristic search strategies are thus more feasible than exhaustive ones.

3. Evaluation strategy. The question how features and feature subsets are evaluated is the single biggest differentiating factor among feature selection algorithms. One paradigm, the so-called feature filter (Kohavi and John, 1997) operates independently of any learning algorithm – undesirable features are filtered out of the data before learning begins. These algorithms use heuristics based on general characteristics of the data to evaluate the potential merit of feature subsets. A different approach suggests that the characteristics of a particular induction algorithm should be taken into account when features are

selected. This method, called the feature wrapper (Kohavi and John, 1997), uses an induction algorithm along with a statistical re-sampling technique such as cross-validation to estimate the final accuracy of feature subsets. A third approach does not treat the learning algorithm as a black box: embedded methods perform feature selection in the process of training and are usually specific to given learning machines (Guyon, Elisseefi, and Kaelbling, 2003).

4. Stopping criterion. A feature selector must decide when to stop searching through the space of feature subsets. Depending on the evaluation strategy, a feature selector might stop adding or removing features when none of the alternatives improves upon the merit of a current feature subset. Alternatively, the algorithm might continue to revise the feature subset as long as the merit does not degrade. A further option could be to continue generating feature subsets until reaching the opposite end of the search space and then select the best.

The heuristic search technique sequential backward elimination was first introduced by (Marill and Green, 1963), while later different variants were proposed, including a forward method and a stepwise method (Kittler, 1978). Searching the space of feature subsets within reasonable time constraints is necessary if a feature selection algorithm is to operate on data with a large number of features, though kernel methods are capable of operating with many relevant features (Cristianini and Shawe-Taylor, 2000). One simple search strategy, called greedy hill climbing, considers local changes to the current feature subset. Often, a local change is simply the addition or deletion of a single feature from the subset. When the algorithm considers only additions to the feature subset it is known as forward selection; considering only deletions is known as backward elimination (Kittler, 1978; Miller, 1990). An alternative approach, called stepwise bidirectional search, uses

both addition and deletion. Within each of these variations, the search algorithm may consider all possible local changes to the current subset and then select the best, or may simply choose the first change that improves the merit of the current feature subset. In either case, once a change is accepted, it is never reconsidered.

Best first search (Rich and Knight, 1983) is a search strategy that allows backtracking along the search path. Like greedy hill climbing, best first moves through the search space by making local changes to the current feature subset. However, unlike hill climbing, if the path being explored begins to look less promising, the best first search can back-track to a more promising previous subset and continue the search from there. Given enough time, a best first search will explore the entire search space, so it is common to use a stopping criterion. Normally this involves limiting the number of fully expanded subsets that result in no improvement.

2.1.1.1 Feature Filters

The earliest approaches to feature selection within machine learning were filter methods. All filter methods use heuristics based on general characteristics of the data, and not a learning algorithm to evaluate the features or feature subsets. As a consequence, filter methods are generally much faster than wrapper methods, and are more practical for use on data of high dimensionality.

Several justifications for the use of filters for subset selection have been put forward (Guyon, Elisseefi, and Kaelbling, 2003). Compared to wrappers, filters are faster, but efficient embedded methods are competitive in that respect. Another argument is that some filters (such as those based on mutual information criteria) provide a generic selection of features, not tuned for a given inductive learner. A further compelling justification is that filtering can be used as a preprocessing step to reduce space dimensionality and overcome overfitting.

Feature Ranking Many feature selection algorithms include feature ranking as a principal or auxiliary selection mechanism because of its simplicity, scalability, and good empirical success. Several papers use feature ranking as a baseline method (Bekkerman et al., 2003; Forman, Guyon, and Elisseeff, 2003; Caruana et al., 2003; Weston et al., 2003). Feature ranking is not necessarily used to build predictors.

Consider a set of N examples $\{x_k, c_k\}$ $(k = 1, \ldots, N)$ consisting of M input features f_i $(i = 1, \ldots, M)$ and one output class c_k. Feature ranking makes use of a scoring function $S(i)$ computed from the values f_i and c_k , $i = 1, \ldots, M$, $k = 1, \ldots, N$. By convention, it is assumed that a high score is indicative of a valuable feature and that features are sorted in decreasing order of $S(i)$. To use feature ranking to build predictors, nested subsets incorporating progressively more and more features of decreasing relevance are defined.

Following the classification of (Kohavi and John, 1997), feature ranking is a filter method: it is a preprocessing step, independent of the choice of the predictor. Nevertheless, under certain independence or orthogonality assumptions, it may be optimal with respect to a given predictor (Guyon, Elisseefi, and Kaelbling, 2003). For instance, using Fisher's criterion to rank features in a classification problem where the covariance matrix is diagonal is the optimum for Fisher's linear discriminant classifier (Duda, Hart, and Stork, 2000). Even when feature ranking is not optimal, it may be preferable to other feature subset selection methods because of its computational and statistical scalability: computationally, it is efficient since it requires only the computation of M scores and sorting the scores; statistically, it is robust against overfitting because it introduces bias but it may have considerably less variance (Hastie, Tibshirani, and Friedman, 2001)

If the input vector x can be interpreted as the realization of a random vector drawn from an underlying unknown distribution, let X_i denote the random feature

corresponding to the ith component of x. Similarly, C will be the random class of which the outcome c is a realization. Further, let $\mathbf{x}_i$ denote the N dimensional vector containing all the realizations of the ith feature for the training examples, and $\mathbf{c}$ the N dimensional vector containing all the target values.

Let us consider first the prediction of a continuous outcome y. The estimation of the Pearson correlation coefficient is defined as:

$$R(i) = \frac{\sum_{k=1}^{N}(x_{ki} - \bar{\mathbf{x}}_i)(c_k - \bar{\mathbf{c}})}{\sqrt{\sum_{k=1}^{N}(x_{ki} - \bar{\mathbf{x}}_i)^2 \sum_{k=1}^{N}(c_k - \bar{\mathbf{c}})^2}},$$

where the bar notation stands for an average over the index k. This coefficient is also the cosine between vectors $\mathbf{x}_i$ and $\mathbf{c}$, after they have been centered (their mean subtracted).

In linear regression, the coefficient of determination, which is the square of $R(i)$, represents the fraction of the total variance around the mean value $\mathbf{c}$ that is explained by the linear relation between $\mathbf{x}_i$ and c. Therefore, using $R(i)^2$ as a feature ranking criterion enforces a ranking according to goodness of linear fit of individual features.

The use of $R(i)^2$ can be extended to the case of two-class classification, for which each class label is mapped to a given value of y, e.g., ± 1. $R(i)^2$ can then be shown to be closely related to Fisher's criterion (Furey et al., 2000), to the T-test criterion, and other similar criteria (Hastie, Tibshirani, and Friedman, 2001)

Correlation criteria such as $R(i)$ can only detect linear dependencies between feature and target. A simple way of lifting this restriction is to make a non-linear fit of the target with single features and rank according to the goodness of fit. Because of the risk of overfitting, one can alternatively consider using non-linear preprocessing (such as squaring, taking the square root, the log, the inverse, etc.) and then using a simple correlation coefficient.

One can extend the classification case the idea of selecting features according to their individual predictive power, using as criterion the performance of a classifier

built with a single feature. For example, the value of the feature itself (or its negative, to account for class polarity) can be used as discriminant. A classifier is obtained by setting a threshold θ on the value of the feature (e.g., at the mid-point between the center of gravity of the two classes). The predictive power of the feature can be measured in terms of error rate. Various other criteria can be defined that involve false positive classification rate fpr and false negative classification rate fnr. The trade-off between fpr and fnr is monitored by varying the threshold θ. ROC curves that plot "hit" rate (1-fpr) as a function of "false alarm" rate "fnr" are essential in defining criteria such as: The "Break Even Point" (the hit rate for a threshold value corresponding to $fpr = fnr$) and the "Area Under Curve" (the area under the ROC curve). In the case where there is a large number of features that separate the data perfectly, ranking criteria based on classification success rate cannot distinguish between the top ranking features. One will then prefer to use a correlation coefficient or another statistic like the margin (the distance between the examples of opposite classes that are closest to one another for a given feature).

Several approaches to the feature selection problem using information theoretic criteria have been proposed (Koller and Sahami, 1996; Bekkerman et al., 2003; Forman, Guyon, and Elisseeff, 2003). As the goal of an induction algorithm is to estimate the probability distributions over the class values given the original feature set, feature subset selection should attempt to remain as close to these original distributions as possible. Many rely on empirical estimates of the mutual information between each feature and the target (Guyon, Elisseefi, and Kaelbling, 2003):

$$I(i) = \int_{\mathbf{x_i}} \int_{y} p(\mathbf{x_i}, c) \log \frac{p(\mathbf{x}_i, c)}{p(\mathbf{x}_i)p(c)} dxdy,$$

where $p(\mathbf{x}_i)$ and $p(c)$ are the probability densities of $\mathbf{x}_i$ and c, and $p(\mathbf{x}_i, c)$ is the joint density. The criterion $I(i)$ is a measure of dependency between the density of feature $\mathbf{x}_i$ and the density of the target c. The difficulty is that the densities

$p(\mathbf{x}_i)$, $p(c)$ and $p(\mathbf{x}_i, c)$ are all unknown and are hard to estimate from data. The probabilities are estimated from frequency counts. Compelling results have been achieved on a benchmark data set using SVMs (Dumais et al., 1998) based on this measure. The justification for classification problems is that the measure of mutual information does not rely on any prediction process, yet it provides a bound on the error rate using any prediction scheme for the given distribution.

Information gain measures the decrease in entropy when the feature is given vs. absent and it is frequently employed as a feature goodness criterion (Yang and Pedersen, 1997). Let $C = \{c_1, c_2, ..., c_{|C|}\}$ denote the set of categories. The information gain of a feature $\mathbf{x}$ is defined to be:

$$G(\mathbf{x}) = -\sum_{i=1}^{|C|} p(c_i) \log p(c_i) + p(\mathbf{x}) \sum_{i=1}^{|C|} p(c_i|\mathbf{x}) \log p(c_i|\mathbf{x}) + p(\bar{\mathbf{x}}) \sum_{i=1}^{|C|} p(c_i|\bar{\mathbf{x}}) \log p(c_i|\bar{\mathbf{x}}).$$

This measure was found to be compelling (Yang and Pedersen, 1997) for text classification.

The case of continuous variables is the hardest to deal with. An information theoretic filter requires features with more than two values to be encoded as binary in order to avoid the bias that entropic measures have toward features with many values. This can greatly increase the number of features in the original data, as well as introducing further dependencies.

More Complex Feature Evaluation

A filtering approach was introduced to feature selection originally designed for Boolean domains that involves a greater degree of search through the feature space (Almuallim and Dietterich, 1991). The Focus algorithm looks for minimal combinations of features that perfectly discriminate among the classes. This is referred to as the "min-features bias". The method begins by looking at each feature in isolation, then turns to pairs of features, triples, and so forth, halting only when it finds a combination that generates pure partitions of the training

set (that is, in which no instances have different classes). There are two main difficulties with Focus (Caruana and Freitag, 1994). Firstly, since Focus is driven to attain consistency on the training data, an exhaustive search may be intractable if many features are needed to attain consistency. Secondly, a strong bias towards consistency can be statistically unwarranted and may lead to overfitting the training data –the algorithm will continue to add features to repair a single inconsistency. The first problem was later addressed (Almuallim and Dietterich, 1992).

An algorithm similar to Focus called LVF was also developed (Liu and Setiono, 1996). Like Focus, LVF is consistency driven and, unlike Focus, can handle noisy domains if the approximate noise level is known a priori. LVF generates a random subset S from the feature subset space during each round of execution. If S contains fewer features than the current best subset, the inconsistency rate of the dimensionally reduced data described by S is compared with the inconsistency rate of the best subset. If S is at least as consistent as the best subset, S replaces the best subset. The inconsistency rate of the training data prescribed by a given feature subset is defined over all groups of matching instances. Within a group of matching instances the inconsistency count is the number of instances in the group minus the number of instances in the group with the most frequent class value. The overall inconsistency rate is the sum of the inconsistency counts of all groups of matching instances divided by the total number of instances. Results for LVF on natural domains were mixed (Liu and Setiono, 1996).

Feature selection based on rough sets theory (Modrzejewski, 1993; Pawlak, 1991) uses notions of consistency similar to those described above. In rough sets theory an information system is a 4-tuple $S = (U, Q, V, f)$, where U is the finite universe of instances, Q is the finite set of features, V is the set of possible feature values, f is the information function: given an instance and a feature, f maps them to a value $v \in V$. For any subset of features $P \subseteq Q$, an indiscernibility relation

IND(P) is defined as:

$$\text{IND}(P) = (x, y) \in U \times U : f(x, a) = f(y, a),$$

for every feature $a \in P$.

The indiscernibility relation is an equivalence relation over U. Hence, it partitions the instances into equivalence classes – sets of instances indiscernible with respect to the features in P. Such a partition (classification) is denoted by $U/\text{IND}(P)$. In supervised machine learning, the sets of instances indiscernible with respect to the class feature contain the instances of each class. For any subset of instances $X \subseteq U$ and subset of features $P \subseteq Q$, the lower $\underline{\text{P}}$, and the upper, $\bar{P}$ approximations of X are defined as follows:

$$\underline{\text{P}} = \cup\{Y \in U/\text{IND}(P) : Y \subseteq X\},$$

$$\bar{P} = \cup\{Y \in U/\text{IND}(P) : Y \cap X \neq \emptyset\}.$$

If $\underline{\text{P}}(X) = \bar{P}(X)$ then X is an exact set (definable using feature subset P), otherwise X is a rough set with respect to P. The instances in U that can be classified to the equivalence classes of $U/\text{IND}(P)$ by using feature set R are called the positive region of P with respect to R, and is defined as follows: $\text{POS}_R = \bigcup_{X \in U/\text{IND}(P)} \underline{\text{R}}(X)$. The degree of consistency afforded by feature subset R with respect to the equivalence classes of $U/\text{IND}(P)$ is given by: $\gamma_R(P) = \frac{|\text{POS}_R(P)|}{|U|}$. If $\gamma_R(P) = 1$ then P is totally consistent with respect to R. Feature selection in rough sets theory is achieved by identifying a reduct of a given set of features. A set $R \subseteq P$ is a reduct of P if it is independent and $\text{IND}(R) = \text{IND}(P)$. R is independent if there does not exist a strict subset R' of R such that $\text{IND}(R') = \text{IND}(R)$. Each reduct has the property that a feature cannot be removed from it without changing the indiscernibility relation. Both rough sets and the LVF algorithm are likely to assign higher consistency to features that have many values.

Using One Learning Algorithm as a Filter for Another

Several researchers have explored the possibility of using a particular learning algorithm as a pre-processor to discover useful feature subsets for a primary learning algorithm.

C4.5 was applied to three natural language data sets; only the features that appeared in the final decision trees were used with a nearest neighbor classifier (Cardie, 1993). The use of this hybrid system resulted in significantly better performance than either C4.5 or the nearest neighbor algorithm when used alone. In a similar approach, (Singh and Provan, 1996) used a greedy oblivious decision tree algorithm to select features from which to construct a Bayesian network. Oblivious decision trees differ from those constructed by algorithms such as C4.5 in that all nodes at the same level of an oblivious decision tree test the same feature. Results showed that Bayesian networks using features selected by the oblivious decision tree algorithms outperformed Bayesian networks without feature selection and even Bayesian networks with features selected by a wrapper.

Decision table majority (DTM) classifiers are a simple type of nearest neighbor classifiers where the similarity function is restricted to returning stored instances that are exact matches with the instance to be classified (Pfahringer, 1995). If no instances are returned, the most prevalent class in the training data is used as the predicted class; otherwise, the majority class of all matching instances is used. Induction of a DTM is achieved by greedily searching the space of possible decision tables. Since a decision table is defined by the features it includes, induction is simply feature selection. The minimum description length (MDL) principle (Rissanen, 1978) guides the search by estimating the cost of encoding a decision table and the training examples it misclassifies with respect to a given feature subset (Pfahringer, 1995). The features appearing in the final decision table are then used with other learning algorithms. Experiments on a small selection of machine

learning data sets showed that feature selection by DTM induction can improve accuracy in some cases.

2.1.1.2 Feature Weighting Algorithms

Feature weighting can be regarded as a generalization of feature selection. In feature selection, a feature is either used or is not, that is, feature weights are restricted to 0 or 1. Feature weighting allows finer differentiation between features by assigning each a continuously valued weight. Algorithms such as nearest neighbor (that normally treat each feature equally) can be easily modified to include feature weighting when calculating similarity between cases. Feature weighting algorithms do not reduce the dimensionality of the data: unless features with very low weights are removed from the data initially, it is assumed that each feature is useful for induction. Its degree of usefulness is reflected in the magnitude of its weight. Using continuous weights for features involves searching a much larger space and involves a greater chance of overfitting (Kohavi, Langley, and Yun, 1997).

An algorithm called Exemplar-Aided Constructor for Hyperreactangles (EACH) incorporates incremental feature weighting in an instance based learner (Salzberg, 1991). For each correct classification made, the weight for each matching feature is incremented so-called the global feature adjustment rate, while mismatching features have their weights decremented by this same amount. For incorrect classifications, the opposite occurs: mismatching features are incremented while the weights of matching features are decremented. The value of the global feature adjustment rate needs to be tuned for different data sets to give best results.

The weighting scheme of EACH is insensitive to skewed concept descriptions (Wettschereck and Aha, 1995) . IB4 is an extension of the k nearest neighbor algorithm that addresses this problem by calculating a separate set of feature weights for each concept (Aha, 1992). Experiments with IB4 showed it to be more tolerant

of irrelevant features than the k nearest neighbor algorithm.

An algorithm called Relief was developed which follows the general paradigm of feature ranking, but incorporates a more complex feature evaluation function (Kira and Rendell, 1992). It uses instance based learning to assign a relevance weight to each feature. Each feature's weight reflects its ability to distinguish among the class values. Features are ranked by weight and those that exceed a user-specified threshold are selected to form the final subset. The algorithm operates by randomly sampling instances from the training data. For each sampled instance, the nearest instance of the same class and opposite class is sought. A feature's weight is updated according to how well its values distinguish the sampled instance from its nearest hit and nearest miss. A feature will receive a high weight if it differentiates between instances from different classes and has the same value for instances of the same class. The following formula is used in updating:

$$W(f_i) = W(f_i) - \frac{\mathrm{diff}(f_i, R, H)^2}{m} + \frac{\mathrm{diff}(f_i, R, M)^2}{m},$$

where $W(f_i)$ is the weight for feature f_i, R is a randomly sampled instance, H is the nearest hit, M is the nearest miss, and m is the number of randomly sampled instances. The function diff calculates the difference between two instances for a given feature. For nominal features it is defined as either 1 (the values are different) or 0 (the values are the same), while for continuous features the difference is the actual difference normalized to the interval $[0, 1]$. Dividing by m guarantees that all weights are in the interval $[-1, 1]$. Relief was found to be effective at identifying relevant features even when they interact (Kira and Rendell, 1992). The original Relief operated on two-class domains, while later it was extended to multi-class, noisy and incomplete domains (Kononenko, 1994). Relief was used to calculate weights for a k nearest neighbor algorithm – significant improvement was reported over standard k nearest neighbor in seven out of ten domains (Wettschereck and Aha, 1995). Relief was also adopted to perform feature selection directly in the

kernel space of a support vector machine (Cao et al., 2007).

Another approach to feature weighting considers a small set of discrete weights rather than continuous weights (Kohavi, Langley, and Yun, 1997). This approach uses the wrapper coupled with simple nearest neighbor to estimate the accuracy of feature weights and a best first search to explore the weight space. In experiments that vary the number of discrete weights considered by the algorithm, results showed that there is no advantage to increasing the number of non-zero discrete weights above two; in fact, with the exception of some carefully crafted artificial domains, using one non-zero weight (equivalent to feature selection) was difficult to outperform.

2.1.1.3 Feature Wrappers

The wrapper methodology offers a simple and powerful way to address the problem of feature selection, regardless of the chosen learning algorithm (Kohavi and John, 1997). The learning algorithm is considered a black box and the method lends itself to the use of off-the-shelf machine learning software packages. In its most general formulation, the wrapper methodology consists in using the prediction performance of a given learning machine to assess the relative usefulness of subsets of features. The use of cross-validation for estimating the accuracy of a feature subset–which is the backbone of the wrapper method in machine learning–was suggested by (Allen, 1974) and applied to the problem of selecting predictors in linear regression. Feature wrappers often achieve better results than filters due to the fact that they are tuned to the specific interaction between an induction algorithm and its training data (Langley, 1994).

Formal definitions for two degrees of feature relevance were formulated, and it was claimed that feature wrappers are able to discover relevant features (John, Kohavi, and Pfleger, 1994). A feature f_i is said to be strongly relevant to the target

concept if the probability distribution of the class values, given the full feature set, changes when f_i is removed. A feature f_i is said to be weakly relevant if it is not strongly relevant and the probability distribution of the class values, given some subset S (containing f_i) of the full feature set, changes when f_i is removed. All features that are not strongly or weakly relevant are irrelevant.

Wrappers are often criticized because they seem to be a "brute force" method requiring massive amounts of computation, but it is not necessarily so. Efficient search strategies may be devised. Using such strategies does not necessarily mean sacrificing prediction performance. In fact, it appears to be the converse in some cases: coarse search strategies may alleviate the problem of overfitting (Reunanen, Guyon, and Elisseeff, 2003). In fact, greedy search strategies seem to be particularly computationally advantageous and robust against overfitting. By using the learning machine as a black box, wrappers are remarkably universal and simple.

Most criticism of the wrapper approach to feature selection is concerned with its computational cost. For each feature subset examined, an induction algorithm is invoked k times in a k-fold cross validation. This can make the wrapper prohibitively slow for use on large data sets with many features. This drawback has led some researchers to investigate ways of mitigating the cost of the evaluation process.

Given the complexity of feature wrappers, it seems reasonable to use a wrapper (or embedded method) with a linear predictor as a filter and then train a more complex non-linear predictor on the resulting features. An example of this approach is found in the paper of (Bi et al., 2003): a linear 1-norm SVM is used for feature selection, but a non-linear 1-norm SVM is used for prediction. The complexity of linear filters can be ramped up by adding to the selection process products of input features (monomials of a polynomial) and retaining the features that are part of any selected monomial.

The wrapper approach was proposed at approximately the same time and independently of (John, Kohavi, and Pfleger, 1994) by (Langley and Sage, 1994a; Langley and Sage, 1994a) during their investigation of the simple nearest neighbor algorithm's sensitivity to irrelevant features. Scaling experiments showed that the nearest neighbor's sample complexity (the number of training examples needed to reach a given accuracy) increases exponentially with the number of irrelevant features present in the data (Aha, Kibler, and Albert, 1991; Langley and Sage, 1994a; Langley and Sage, 1994a). A similar approach to augmenting nearest neighbor algorithms was developed using leave-one-out instead of k-fold cross-validation and concentrates on improving the prediction of numeric rather than discrete classes (Moore and Lee, 1994). Using leave-one-out cross validation and a beam search instead of hill climbing (Aha and Bankert, 1994), results show that feature selection can improve the performance of a nearest neighbor classifier on a sparse cloud pattern domain with many features.

A context sensitive wrapper approach to feature selection was also developed (Domingos, 1997). The motivation for the approach is that there may be features that are either relevant in only a restricted area of the instance space and irrelevant elsewhere, or relevant given only certain values (weakly interacting) of other features and otherwise irrelevant. In either case, when features are estimated globally (over the entire instance space), the irrelevant aspects of these sorts of features may overwhelm their useful aspects for instance based learners. This is true even when using backward search strategies with the wrapper. The algorithm called RC which can detect and make use of context sensitive features works by selecting a (potentially) different set of features for each instance in the training set. It does this by using a backward search strategy and cross validation to estimate accuracy. For each instance in the training set, RC finds its nearest neighbor of the same class and removes those features in which the two differ. The accuracy of

the entire training data set is then estimated by cross validation. If the accuracy has not degraded, the modified instance in question is accepted; otherwise the instance is restored to its original state and deactivated (no further feature selection is attempted for it). The feature selection process continues until all instances are inactive. Experiments on a selection of machine learning data sets showed that RC outperformed standard wrapper feature selectors using forward and backward search strategies with instance based learners.

Embedded methods that incorporate feature selection as part of the training process may be more efficient in several respects: they make better use of the available data by not needing to split the training data into a training and validation set; they reach a solution faster by avoiding retraining a predictor from scratch for every feature subset investigated. Embedded methods are not new: decision trees such as CART, for instance, have a built-in mechanism to perform feature selection (Breiman et al., 1984).

A number of learning machines extract features as part of the learning process. These include neural networks whose internal nodes are feature extractors. Thus, node pruning techniques such as OBD (Le Cun, Denker, and Solla, 1990) are feature selection algorithms. Gram-Schmidt orthogonalization was also applied as an alternative to OBD (Stoppiglia et al., 2003).

Kernel methods possess an implicit feature space revealed by the kernel expansion: $K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x}), \phi(\mathbf{x}')$, where $\phi(\mathbf{x})$ is a feature vector of possibly infinite dimension. Selecting these implicit features may improve generalization, but does not change the running time or help interpreting the prediction function. A method for selecting implicit kernel features in the case of the polynomial kernel, using their framework of minimization of the 0-norm, was proposed (Weston et al., 2003).

2.1.2 Feature Construction and Space Dimensionality Reduction

In some applications, reducing the dimensionality of the data by selecting a subset of the original features may be advantageous for reasons including the expense of making, storing and processing measurements. If these considerations are not of concern, other means of space dimensionality reduction should also be considered. There are a number of generic feature construction methods, including: clustering; basic linear transforms of the input features (PCA/SVD, LDA); more sophisticated linear transforms like spectral transforms (Fourier, Hadamard), wavelet transforms or convolutions of kernels; and applying simple functions to subsets of features, like products to create monomials.

Two distinct goals may be pursued for feature construction: achieving best reconstruction of the data or being most efficient for making predictions. The first problem is an unsupervised learning problem. It is closely related to that of data compression and a lot of algorithms are used across both fields. The second problem is a supervised one.

2.1.2.1 Clustering

Clustering has long been used for feature construction. The idea is to replace a group of "similar" features by a cluster centroid, which becomes a feature. The most popular algorithms include k means and hierarchical clustering (Jain, Murty, and Flynn, 1999).

The k nearest neighbor algorithm (also referred to as k means algorithm) is a method to cluster objects based on features into k partitions. It tries to minimize overall intra-cluster variance. The algorithm starts by partitioning the input points into k initial sets, either at random or using some heuristic data. It then calculates

the centroid, of each set as follows.

$$\mathbf{c} = \frac{1}{M_c} \sum_{j=1}^{M_c} \mathbf{x}_j,$$

where M_c is the number of vectors in the subset. It constructs a new partition by associating each point with the closest centroid. The centroid-object distances are computed by the usual cosine dissimilarity. Then the centroids are recalculated for the new clusters, and the process is repeated by alternate application of these two steps until convergence, which is obtained when the points no longer switch clusters. The algorithm does not guarantee a global optimum for clustering. The quality of the final solution depends largely on the initial set of clusters, however, in text classification it has been proved to be efficient (Steinbach, Karypis, and Kumar, 2000).

The construction of a k nearest neighbor classifier involves determining a threshold k indicating how many top-ranked training objects have to be considered. (Larkey and Croft, 1996) used $k = 20$, while others found $30 \leq k \leq 45$ to yield the best effectiveness (Yang and Chute, 1994; Joachims, 1998; Yang and Liu, 1999).

k nearest neighbor does not build an explicit, declarative representation of the category c_i , but it has to rely on the category labels attached to the training objects similar to the test objects. k nearest neighbor makes a prediction based on the k training patterns that are closest to the unlabeled example. For deciding whether $\mathbf{x}_j \in c_k$, k nearest neighbor looks at whether the k training objects most similar to $\mathbf{x}_j$ also are in c_k ; if the answer is positive for a large enough proportion of them, a positive decision is taken (Yang and Chute, 1994).

Unlike linear classifiers, k nearest neighbor does not divide the object space linearly, hence it tends to perform better on linearly inseparable problems (Steinbach, Karypis, and Kumar, 2000). The most significant disadvantage is its inefficiency at classification time. Linear classifiers consider a simple dot product, while

k NN requires the entire training objects to be ranked for similarity with the test objects (Lan et al., 2009).

The CB-SVM algorithm handles very large data sets by condensing the training data into the statistical summaries of large data groups such that coarse summary is made for "unimportant" data and fine summary is made for "important" data (Yu, Yang, and Han, 2003).

Given n M-dimensional data points in a cluster: $\{\mathbf{x}_i\}$ where $i = 1, 2, \ldots, n$, the centroid C and the radius R of the clusters are defined as:

$$C = \frac{\sum_{i=1}^{n} x_i}{n} \tag{2.1}$$

$$R = \left(\frac{\sum_{i=1}^{n} ||x_i - C||^2}{n} \right)^{\frac{1}{2}} \tag{2.2}$$

R is the average distance from member points to the centroid.

The concepts of clustering feature (CF) tree is at the core of the hierarchical micro-clustering algorithm which makes the clustering incremental without expensive computations. A CF is a triple which summarizes the information that a CF tree maintains for a cluster (Zhang, Ramakrishnan, and Livny, 1996; Yu, Yang, and Han, 2003).

Given n M-dimensional data points in a cluster: $\{\mathbf{x}_i\}$ where $i = 1, 2, \ldots, n$, the CF vector of the cluster if defined as a triple: $CF = (n, LS, SS)$, where n is the number of data points in the cluster, LS is the linear sum of the n data points, i.e., $\sum_{i=1}^{n} x_i$, and SS is the square sum of the n data points, i.e., $\sum_{i=1}^{n} x_i^2$.

Assume that CF_1=(n_1, LS_1, SS_1) and $CF_2 = (n_2, LS_2, SS_2)$ are the CF vectors of two disjoint clusters. Then the CF vector of the cluster that is formed by merging the two disjoint clusters is CF_1+$CF_2 = (n_1$+n_2, LS_1+LS_2, SS_1+$SS_2)$. See (Zhang, Ramakrishnan, and Livny, 1996) for a proof. From the above definition and theorem, it is known that the CF vectors of clusters can be stored and calculated incrementally and accurately as clusters are merged. The centroid C and the radius

R of each cluster can be also computed from the CF of the cluster.

This CF tree is a compact representation of the data set because each entry in a leaf node is not a single data point but a subcluster (which absorbs many data points with radius under a specific threshold t). Managing this CF summary is efficient, saves space significantly, and is sufficient for calculating all the information for building the hierarchical micro-cluster which will facilitate computing an SVM boundary for very large data sets.

A CF tree is a height balanced tree with two parameters: branching factor b and threshold t. Each nonleaf node consists of at most b entries of the form $(CF_i, child_i)$, where $i = 1, 2, \ldots, b$, $child_i$ is a pointer to its i-th child node, and CF_i is the CF of the subcluster represented by this child. A leaf entry, the entry in a leaf node, only has a CF without a child pointer. So, a leaf or a nonleaf node represents a cluster made up of all the subclusters represented by its entries. The threshold t is a constraint for the leaf entries to satisfy such that the radius of an entry has to be less than t. The tree size is a function of t. The larger the t is, the smaller the tree is. The branching factor b can be determined by a page size such that a leaf or nonleaf node fit in a page.

A CF tree is built up dynamically as new data objects are inserted. The ways that it inserts a data into the correct subcluster, merges leaf nodes and manages nonleaf nodes are similar to those in a B+-tree, which can be sketched as follows:

1. Identifying the appropriate leaf: Starting from the root, it descends the CF tree by choosing the child node whose centroid is the closest.

2. Modifying the leaf: If the leaf entry can absorb the new data object without violating the threshold condition, updates just the CF vector of the entry. If not, add a new entry. If adding a new entry causes a node split, split by choosing the farthest pair of entries as seeds, and redistributing the remaining entries based on the closest criteria.

3. Modifying the path to the leaf: It updates the CF vectors of each nonleaf entry on the path to the leaf. Node split in the leaf causes an insertion of a new nonleaf entry into the parent node, and if the parent node becomes split, a new entry is inserted into the higher level node. Likewise, this occurs recursively to the root.

Any clustering algorithm can be used with CF trees (Zhang, Ramakrishnan, and Livny, 1996). The CPU and I/O costs of the BIRCH algorithm are of order $O(n)$. A number of experiments reported linear scalability of the algorithm with respect to the number of points, insensibility to the input order, and good quality of clustering of the data (Zhang, Ramakrishnan, and Livny, 1996).

The key idea of cluster-based SVMs (CBSVMs) can be viewed being similar to that of selective sampling, that is, selecting the data which maximizes the benefit of learning. The selective sampling for SVMs selects and accumulates the low margin data at each round that are close to the boundary in the feature space because the low margin data have higher chances to become the support vectors of the boundary for the next round (Schohn and Cohn, 2000; Tong, Koller, and Kaelbling, 2001). Following this idea, the original algorithm declusters the entries near the boundary to get finer samples nearer to the boundary and coarser samples farther from the boundary (Yu, Yang, and Han, 2003). In this way, the support vectors, the description of the boundary, are as fine as possible while keeping the total number of training data points as small as possible. The outline of the CBSVM algorithm follows:

1. Construct two CF trees from positive and negative data sets independently.

2. Train an SVM boundary function from the centroids of the root entries – entries in the root node– of the two CF trees. If the root node contains too few entries, train from the entries of the nodes in the second levels of the trees.

3. Decluster the entries near the boundary into the next level, and the children entries declustered from the parent entries are accumulated into the training set with the non-declustered parent entries.

4. Construct another SVM from the centroids of the entries in the training set, and repeat from step 3 until nothing is accumulated.

2.1.2.2 Matrix Factorization

Another widely used method of feature construction is singular value decomposition (SVD). The goal of SVD is to form a set of features that are linear combinations of the original features, which provide the best possible reconstruction of the original data in the least square sense (Duda, Hart, and Stork, 2000). It is an unsupervised method of feature construction.

Let the real-valued M by N matrix X be a linear transformation from the N dimensional Euclidean space E_N to the M dimensional Euclidean space E_M in two arbitrarily chosen bases of these spaces. The singular values of X are gained by the eigen base of X, where the eigen base of X is an orthonormal base of the E_N space, and

$$(Xu_i, Xu_j) = \begin{cases} \sigma_i^2 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

The left-hand side is a scalar product of the E_M space. The σ_i values are the singular values. Let U denote the set of u_i vectors, this is the (left-hand side) eigen base of X. Any linear transformation has an eigen base, which is not unique, but the singular values are unique (apart from the order). Let $v_i = Xu_i/\sigma_i$ $(i = 1, 2, \ldots, r)$, where X^* is the adjoint of X and r is the rank of X. For zero singular values v_i can be chosen arbitrarily, but the resulting V set of v_i vectors should be orthonormal.

If the bases in the E_N and E_M spaces are fixed, the matrix of U is $M \times r$, if those vectors which belong to zero singular values are omitted. The matrix of V

is $r \times N$, with the same condition. Let Σ denote a rectangular matrix, its diagonal consisting of the singular values, the other elements are zero. By the orthogonality of U and V, the following decomposition is derived:

$$X = (UU^*)X = (u_1u_1^* + u_2u_2^* + \cdots + u_ru_r^*)X = \tag{2.3}$$

$$\sigma_1u_1v_1^* + \sigma_2u_2v_2^* + \cdots + \sigma_ru_rv_r^* = U\Sigma V^*.$$

The above formula is the singular value decomposition of the matrix X. Let Σ_k denote that matrix which is similar to Σ, but it has only the k highest singular values in its diagonal. Then

$$X_k = U\Sigma_k V^*. \tag{2.4}$$

Theorem.(Eckart and Young) Let the SVD of X be given by Equation 2.3 with $r = \text{rank}(X) \leq p = \min(M, N)$, and X_k be given by Equation 2.4, then

$$\min_{\text{rank}(Y)=k} \|X - Y\|_F^2 = \|X - X_k\|_F^2 = \sigma_{k+1}^2 + \cdots + \sigma_p^2.$$

(See (Golub and Reinsch, 1971) for a proof).

X_k is the best approximation to X for any unitarily invariant norm (Berry, Dumais, and O'Brien, 1995; Mirsky, 1960), hence X_k is the closest rank k matrix to X in the sense of the least squares' method as well.

In a similar vein to singular value decomposition, an information theoretic unsupervised feature construction method, the sufficient dimensionality reduction was introduced (Globerson et al., 2003). The most informative features are extracted by solving an optimization problem that monitors the trade-off between data reconstruction and data compression, similar to the information bottleneck of (Tishby, Pereira, and Bialek, 2000); the features are found as Lagrange multipliers of the objective optimized. Non-negative matrices P of dimension (M, N) representing the joint distribution of two random features (for instance the co-occurrence of words in documents) are considered. The features are extracted by information

theoretic I-projections, yielding a reconstructed matrix of special exponential form $\tilde{P} = (1/Z)\exp(\Phi\Psi)$. For a set of d features, Φ is a $(M, d+2)$ matrix whose $(d+1)$th column is ones and Ψ is a $(d+2, n)$ matrix whose $(d+2)$th column is ones, and Z is a normalization coefficient. Similarly to SVD, the solution shows the symmetry of the problem with respect to patterns and features.

2.2 Supervised Machine Learning for Classification

Categorization is the task of assigning unlabeled objects into predefined categories. Given a collection of $\{x_1, x_2, \ldots, x_N\}$ objects, and a $C = \{c_1, c_2, ..., c_{|C|}\}$ set of predefined categories, the task is, for each object x_j ($j \in \{1, 2, \ldots, N\}$), to assign a decision to file x_j under c_i or a decision not to file x_j under c_i ($c_i \in C$) by virtue of a function Φ, where the function Φ is also referred to as the classifier, or model, or hypothesis, or rule. Supervised classification is a machine learning technique for creating the function Φ from training data. The training data consist of pairs of input objects, and desired outputs (i.e., classes).

Following (Sebastiani, 2002), this thesis assumes that:

1. The category label is just symbolic, and no additional knowledge of their meaning is available.

2. Any metadata (or exogenous knowledge, such as, publication date, object type, publication source, etc.) are discarded.

The rest of this section briefly reviews the most common supervised classification methods. Support vector machines have been found the most effective by several authors and the proposed classification method is grounded in the kernel methods underlying support vector machines, hence this family of algorithms is discussed in more detail.

2.2.1 Naïve Bayes Classifier

Probabilistic classifiers learn the classifier function in terms of $p(c|x)$, that is, the probability that an object x belongs to a class c , and estimate this conditional probability by using Bayes' theorem (Lewis, 1998):

$$p(c|x) = \frac{p(x|c)p(c)}{p(x)}$$

Since only the relative order of the category probabilities is important, and by definition, $p(x)$ is independent of c, the above probability can be replaced:

$$p(c|x) \approx p(x|c)p(c).$$

By chain rule

$$p(x|c) = \prod_{i=1}^{M} p(f_i|f_1, f_2, \ldots, f_{i-1}, c).$$

Using the naïve Bayes assumption, each feature of an object is independent of its context, that is

$$p(f_i|f_1, f_2, \ldots, f_{i-1}, c) = p(f_i|c),$$

therefore

$$p(x|c) = \prod_{i=1}^{M} p(f_i|c).$$

To estimate $p(c|x)$, one needs to estimate $p(f|c)$ and $p(c)$ for all features f and for all categories $c \in C$. The following estimator is commonly used (Manning and Schütze, 1999):

$$\hat{p}(c) = \frac{N(c)}{\sum_{c \in C} N(x, c)},$$

where $N(c)$ is the number of training objects in category c, and $N(x, c)$ is the number of documents in category c. Similarly, the conditional probabilities of features in c are estimated by

$$\hat{p}(f|c) = \frac{N(c, f)}{\sum_f N(c, f)} = \frac{\sum_{x \in X_c} N(x, f)}{\sum_f \sum_{x \in X_c} N(x, f)},$$

where X_c is the set of objects belonging to class c in the training set, and $N(x, f)$ is the number of occurrences of the feature f in object x.

Using these estimators and the above equations, $p(c|x)$ can be estimated:

$$\hat{p}(c|x) \approx \prod_{i=1}^{M} \hat{p}(f_i|c)\hat{p}(c) = \prod_{i=1}^{M} \hat{p}(f_i, c).$$

2.2.2 Maximum Entropy Models

Maximum entropy is a general technique for estimating probability distributions from data (Csiszár, 1996) widely used for a variety of natural language tasks, such as language modeling, part-of-speech tagging, text segmentation, and text classification (Berger, Della Pietra, and Della Pietra, 1996; Ratnaparkhi, 1998; Nigam, Lafferty, and McCallum, 1999). The underlying principle of maximum entropy is that without external knowledge, one should prefer distributions that are uniform. Labeled training data is used to derive a set of constraints for the model that characterizes the class-specific expectations for the distribution. Constraints are represented as expected values of "features", any real-valued function of an example.

In classification, maximum entropy estimates the conditional distribution of the class label given an object (Nigam, Lafferty, and McCallum, 1999). An object is represented by a vector of features. The expected value of the feature counts is estimated by the labeled training data on a class-by-class basis.

Let any real-valued function of the object and the class be a feature, $f_i(x, c)$. Maximum entropy allows to restrict the model distribution to have the same expected value for this feature as seen in the training data. The learned conditional distribution $p(c|x)$ must have the property (Nigam, Lafferty, and McCallum, 1999):

$$\frac{1}{N} \sum_{j=1}^{N} f_i(x_j, c(x_j)) = \sum_{j=1}^{N} p(x_j) \sum_{c \in C} p(c|x) f_i(x, c).$$

The object distribution $p(x)$ is unknown, it is not important to model it, thus training data are used without class labels as an approximation to the object

distribution, and hence the constraint:

$$\frac{1}{N}\sum_{j=1}^{N} f_i(x_j, c(x_j)) = \frac{1}{N}\sum_{j=1}^{N}\sum_{c\in C} p(c|x) f_i(x, c).$$

When using maximum entropy, the first step is to identify a set of feature functions that are considered useful for classification. Then, for each of these features, measure its expected value over the training data and take this to be a constraint for the model distribution (Nigam, Lafferty, and McCallum, 1999).

When constraints are estimated, a unique distribution exists that has maximum entropy (Csiszár, 1996), and the distribution is always of the exponential form (Della Pietra, Della Pietra, and Lafferty, 1997):

$$p(c|x) = \frac{1}{Z(x)} \exp(\sum_i \lambda_i f_i(x, c)),$$

where λ_i is a parameter to be estimated and $Z(x)$ is simply the normalizing factor to ensure a proper probability:

$$Z(x) = \sum_{c\in C} \exp(\sum_i \lambda_i f_i(x, c)).$$

The solution to the maximum entropy problem is also the solution to a dual maximum likelihood problem for models of the same exponential form, if the constraints are estimated from labeled training data. The likelihood problem has a single global maximum and no local maxima, hence one approach for finding the maximum entropy solution is to guess any initial exponential distribution of the above correct form as a starting point; then follow the gradient to find a maximum. Since there are no local maxima, this will converge to the global maximum likelihood solution for exponential models, which is also the global maximum entropy solution.

For classification with maximum entropy, counts are used as features (Nigam,

Lafferty, and McCallum, 1999):

$$f_{f_i,c'}(x_j, c) = \begin{cases} 0 & \text{if } c \neq c' \\ \frac{x_{ij}}{N(x_j)} & \text{Otherwise} \end{cases}$$

where $N(x)$ is the total number of features in object x.

With this representation, if a word occurs often in one class, one would expect the weight for that word-class pair to be higher than for the word paired with other classes.

Maximum entropy models do suffer from any independence assumptions. For instance, Naïve Bayes would double count the evidence of the phrase "Buenos Aires", in which the two words almost always co-occur, while maximum entropy discounts the λ_i for each of these features such that their weight towards classification is appropriately reduced by half (Nigam, Lafferty, and McCallum, 1999). As a consequence, bigrams and phrases can be easily added as features by maximum entropy, without the fear that features would be overlapping (Nigam, Lafferty, and McCallum, 1999).

2.2.3 Decision Tree

A decision tree classifier (Mitchell, 1997) is a tree in which internal nodes are labeled by features, branches departing from them are labeled by tests on the weight that the feature has in the test object, and leafs are labeled by categories. The classifier categorizes an object x_j by recursively testing for the weights that the features labeling the internal nodes have in vector $\mathbf{x}_j$, until a leaf node is reached; the label of this node is then assigned to x_j (Sebastiani, 2002) .

A method for learning a decision tree for category c_k consists of the following "divide and conquer" strategy (Lan et al., 2009):

- Check whether all the training examples have the same label;

- If not, choose a feature f_i , partition the training objects into classes of objects that have the same value for f_i , and place each such class in a separate subtree.

The above process is recursively repeated on the subtrees until each leaf of the tree generated contains training examples assigned to the same category c_k. The selection of the feature f_i on which to operate the partition is generally made according to an information gain (Lewis and Ringuette, 1994; Cohen and Singer, 1996) or entropy criterion (Sebastiani, 2002). A "fully grown" tree is prone to overfitting, as some branches may be too specific for the training data. Therefore decision tree methods normally include a method for growing the tree and one for pruning it, thus removing the overly specific branches (Mitchell, 1997).

Probabilistic methods are quantitative in nature, and as such have been criticized: despite their effectiveness they are not easily interpretable by humans. Decision trees do not suffer from this problem, as they are symbolic (i.e., nonnumeric) algorithms, and the decision rules are easy to interpret (Sebastiani, 2002).

2.2.4 Rocchio Method

The Rocchio method is used for inducing linear, profile-style classifiers (Sebastiani, 2002). It is an adaptation to text classification of Rocchio's formula for relevance feedback in the vector space model (Rocchio, 1971; Buckley, Salton, and Allan, 1994).

Rocchio's method computes a classifier $(w_{1k}, \ldots, w_{Mk})$ for category c_k by the formula:

$$w_{ik} = \beta \sum_{x_j \in \mathrm{POS}_k} \frac{x_{ij}}{|\mathrm{POS}_k|} - \gamma \sum_{x_j \in \mathrm{NEG}_k} \frac{x_{ij}}{|\mathrm{NEG}_k|},$$

where $\mathrm{POS}_k = \{x_j | \Phi(x_j, c_k) = \text{true}\}$, and $\mathrm{NEG}_k = \{x_j | \Phi(x_j, c_k) = \text{false}\}$. In the above formula, β and γ are control parameters that allow setting the relative

importance of positive and negative examples. For instance, if β is set to 1 and γ to 0 (Hull, 1994; Schutze, Hull, and Pedersen, 1995; Joachims, 1998; Dumais et al., 1998), c_i is the centroid of its positive training examples.

A classifier built by means of the Rocchio method rewards the closeness of a test object to the centroid of the positive training examples, and its distance from the centroid of the negative training examples. By setting β to a high value and γ to a low one, the importance of negative examples is reduced (e.g., (Buckley, Salton, and Allan, 1994; Cohen and Singer, 1996; Joachims, 1997; Wittek, 2006) use $\beta = 16$ and $\gamma = 4$).

The method is easy to implement, and is also quite efficient, since learning a classifier is essentially averaging weights (Sebastiani, 2002). In features of effectiveness, if the objects in a category occur in disjoint clusters (for instance, a set of newspaper articles labeled with the sports category and dealing with different sports), a Rocchio classifier may miss many of them, as the centroid of these objects may fall outside all of these clusters (Sebastiani, 2002).

2.2.5 Neural Networks

A neural network is a network of units, some of which are designated as input and output units. As a classifier, input units represent features, the output unit(s) represent the category or categories, and the weights on the edges connecting units represent dependence relations (Schutze, Hull, and Pedersen, 1995).

For classifying a test object x_j, its feature weights x_{kj} are loaded into the input units; the activation of these units is propagated forward through the network, and the value of the output unit(s) determines the categorization decision(s). Neural networks are trained by backpropagation: the activation of each input pattern is propagated forward through the network, and the error produced is then backpropagated and the parameters changed to reduce the error (Rumelhart, Hinton,

and Williams, 1986).

The simplest type of linear neural network classifier is the perceptron (Dagan, Karov, and Roth, 1997; Ng, Goh, and Low, 1997). Other types of linear neural network classifiers implementing a form of logistic regression have also been proposed and tested, yielding very good effectiveness (Schutze, Hull, and Pedersen, 1995; Wiener, Pedersen, and Weigend, 1995).

A nonlinear neural network (Lam and Lee, 1999; Schutze, Hull, and Pedersen, 1995; Wiener, Pedersen, and Weigend, 1995; Ruiz and Srinivasan, 1999; Weigend, Wiener, and Pedersen, 1999; Yang and Liu, 1999) is a network with one or more additional layers of nodes or neurons, all neurons are with a nonlinear activation function, such as the sigmoid function. This network layout in classification enables higher-order interactions between features that the network is able to learn. The number of hidden units in the neural network affects the generalization performance (Rumelhart, Widrow, and Lehr, 1994). When compared to linear networks, they yielded either no improvement (Schutze, Hull, and Pedersen, 1995) or very small improvements (Wiener, Pedersen, and Weigend, 1995).

2.2.6 Support Vector Machines

Support vector machines are a kind of supervised learning algorithms which were originally developed for linear classification, and later extended to nonlinear classification by the so-called kernel trick. Support vector machines simultaneously minimize the classification error and maximize the geometric margin, they are also referred to as maximum margin classifiers. By maximizing the margin, support vector machines maximize the generalization, and they also support nonlinear separation using advanced kernels (Vapnik, 1998; Burges, 1998).

In its simplest, linear form, a support vector machine is a hyperplane that separates a set of positive examples from a set of negative examples with maximum

margin (Shawe-Taylor and Cristianini, 2004). The formula for the output of a linear support vector machine is $y_i := \text{sgn}((\mathbf{w}, \mathbf{x}_i)+b)$, where $\mathbf{x}_i$ is the ith training example, y_i is the output of the support vector machine for the ith training example, $\mathbf{w}$ is the normal vector to the hyperplane, and parameter b helps determine the offset of the hyperplane the origin (Müller et al., 2001). In the linear case, the margin is defined by the distance of the hyperplane to the nearest of the positive and negative examples (Figure 2.1). Support vectors are the training data that lie on the margin.

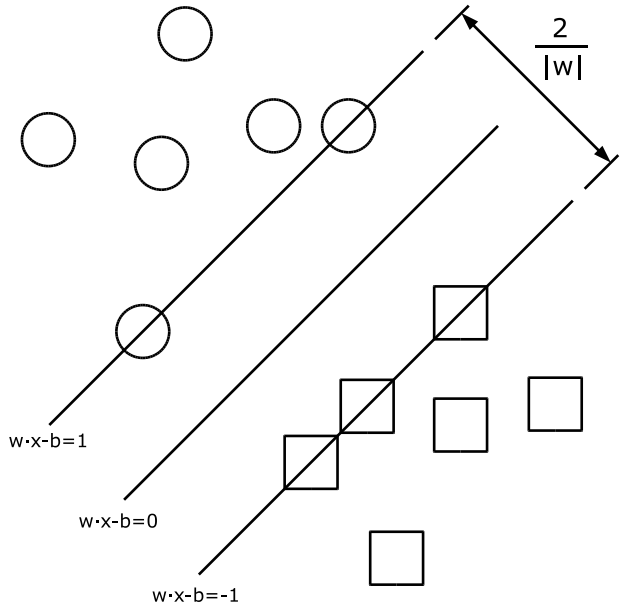


Figure 2.1: Maximal margin hyperplane separating two classes.

The most important kernels are listed in Table 2.1. A sigmoid kernel is also frequently cited ($\tanh(\gamma(\mathbf{x}, \mathbf{y}) + c)$), but this kernel is not positive definite for all choice of parameters, and for the valid range of parameters it was shown to be equivalent with the RBF kernel (Lin and Lin, 2003).

	Linear	Polynomial	RBF
$K(\mathbf{x}, \mathbf{y})$	$(\mathbf{x}, \mathbf{y})$	$((\mathbf{x}, \mathbf{y}) + c)^d$	$\exp(-\gamma\|\mathbf{x} - \mathbf{y}\|^2))$

Table 2.1: Common kernels

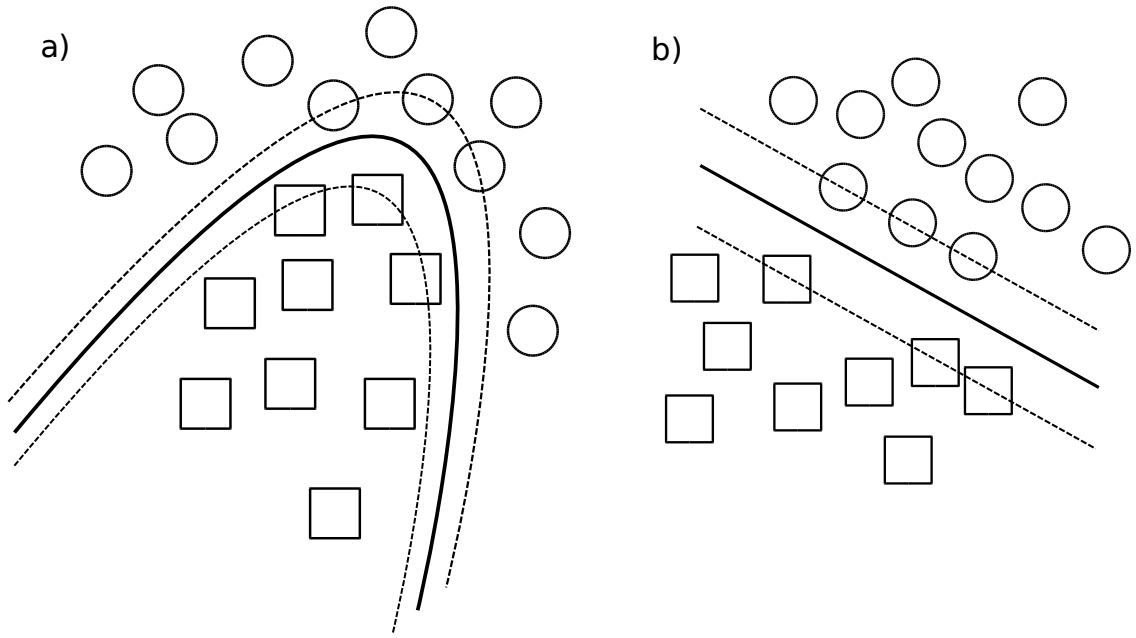


Figure 2.2: The kernel trick. a) Linearly inseparable classification problem. b) The same problem is linearly separable after embedding into a feature space by a nonlinear map ϕ.

The condition for soft-margin classification with support vector machines is

$$\mathbf{w}'\phi(\mathbf{x}_i) + b \geq 1 - \xi_i \quad \text{if } y_i = +1,$$

$$\mathbf{w}'\phi(\mathbf{x}_i) + b \leq -1 + \xi_i \quad \text{if } y_i = -1,$$

where $\phi(.)$ maps $\mathbf{x}$ to a higher dimensional space. The goal of learning is to find $\mathbf{w}$ and b. The primal form of the optimization is to minimize $|\mathbf{w}|$ subject to the above constraints:

$$\min \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i,$$

where C is a constant penalty parameter.

Practically the dual form of the problem is solved, whose number of variables is the same as the number of training examples. The dual form is the following (Burges, 1998):

$$\max_{\alpha} \sum_{i} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i)' \phi(\mathbf{x}_j) =$$

$$\max_{\alpha} \sum_{i} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j), \tag{2.5}$$

subject to

$$C \geq \alpha_i \geq 0, i = 1, \ldots, n,$$

$$\sum_{i} \alpha_i y_i = 0.$$

The solution is given by

$$\mathbf{w} = \sum_{i} \alpha_i y_i \phi(\mathbf{x}_i).$$

The above formulation of support vector machines was originally designed for binary classification. Currently there are two types of approaches for multiclass SVM. One is by constructing and combining several binary classifiers while the other is by directly considering all data in one optimization formulation. The formulation to solve multiclass SVM problems in one step has variables proportional to the number of classes. Therefore, for multiclass SVM methods, either several binary classifiers have to be constructed or a larger optimization problem is needed. Hence in general it is computationally more expensive to solve a multiclass problem than a binary problem with the same number of data (Hsu and Lin, 2002a).

The earliest used implementation for multiclass classification is probably the one-against-all method. It constructs k models where k is the number of classes. The ith SVM is trained with all of the examples in the ith class with positive labels, and all other examples with negative labels. Thus given training

data $(x_1, y_1), \ldots, (x_n, y_n)$, the ith model solves the following problem:

$$\min_{\mathbf{w}^i, b^i, \xi^i} \frac{1}{2}(\mathbf{w}^i)'\mathbf{w}^i + C\sum_{j=1}^{n} \xi_j^i$$

$$(\mathbf{w}^i)'\phi(\mathbf{x}_j) + b^i \geq 1 - \xi_j^i \quad \text{if } y_i = i,$$

$$(\mathbf{w}^i)'\phi(\mathbf{x}_j) + b^i \leq -1 + \xi_j^i \quad \text{if } y_i \neq i,$$

$$\xi_j^i \geq 0, \quad j = 1, \ldots, n.$$

After solving the above problem, there are k decision functions:

$$(\mathbf{w}^1)'\phi(\mathbf{x}) + b^1$$

$$\vdots$$

$$(\mathbf{w}^k)'\phi(\mathbf{x}) + b^k.$$

An $\mathbf{x}$ is in the class which has the largest value of the decision function:

$$\text{class of } \mathbf{x} = \arg\max_{i=1,\ldots,k} (\mathbf{w}^i)'\phi(\mathbf{x}) + b^i.$$

By solving the dual problem, k n-variable quadratic problems are to be solved.

Another major method is the one-against-one method. This method constructs $k(k-1)/2$ classifiers where each one is trained on data from two classes. For training data from the ith and the jth classes, the following binary classification problem is solved:

$$\min_{\mathbf{w}^{ij}, b^{ij}, \xi^{ij}} \frac{1}{2}(\mathbf{w}^{ij})'\mathbf{w}^{ij} + C\sum_{t} \xi_t^{ij}$$

$$(\mathbf{w}^{ij})'\phi(\mathbf{x}_t) + b^{ij} \geq 1 - \xi_t^{ij} \quad \text{if } y_t = i,$$

$$(\mathbf{w}^{ij})'\phi(\mathbf{x}_t) + b^{ij} \leq -1 + \xi_t^{ij} \quad \text{if } y_t = j,$$

$$\xi_t^{ij} \geq 0, \quad \forall t.$$

Thus $k(k-1)/2$ classifiers are constructed.

The one-against-all and one-against-one methods were shown to be superior to the formulation to solve multiclass SVM problems in one step, because the latter approach tends to overfit the training data (Hsu and Lin, 2002a).

2.3 Summary

Among many machine learning algorithms, support vector machines are remarkable for their treatment of nonlinear problems and overall good generalization properties. Since the theory of support vector machines traces back its roots to functional analysis, and not to probability theory, its assumptions are not probabilistic, and constraints only observed on the kernel function. As long as the kernel function is positive definite, the optimization will be a convex problem, hence a global optimum will be found. This is crucial to our argument: nothing is assumed about the features of a vector, nor about the dependencies between the features.

Feature selection and weighting methods assist learning algorithms in various ways. Some learning methods require independent features, and one may observe performance improvements in others. The interplay between feature engineering and kernel methods is not widely researched, and this gives the context to the present work: the purpose is to utilize the nonlinear nature of kernel methods to leverage on feature interdependency to achieve even better generalization performance on classification problems.

Chapter 3

Kernels in the L_2 Space

This chapter introduces a kernel type in the infinite dimensional L_2 space which expands the feature set of the input objects, weights the expansion features, and this weighting is different for each individual input object.

Wavelet kernels have been introduced for both support vector regression and classification (Section 3.1). Most of these wavelet kernels do not use the inner product of the embedding space, but use wavelets in a similar fashion to radial basis function kernels (Section 3.1.4). Interpreting the vector representation of an object as a series of equally spaced observations of a hypothetical continuous signal and approximating the signal with compactly supported basis functions (CSBF) and employing the inner product of the embedding L_2 space, we gain a new family of wavelet kernels. Section 3.2 introduces these CSBF kernels. The proposed kernels are mathematically valid (Section 3.3) and their computational complexity is comparable to that of a linear kernel (Section 3.4). Wavelet analysis is typically carried out on data with a temporal or spatial relation between consecutive data points (Sections 3.1.1, 3.1.2 and 3.1.3). Section 3.5 argues that it is possible to order the features of a general data set so that consecutive features are statistically related to each other, thus enabling the effective use of wavelet kernels on a more general

family of data sets. The CSBF kernels can be implemented efficiently with existing libraries (Section 3.6).

Using a quantitative methodology (Section 3.7), experimental results (Section 3.8) show that with few relevant features and many irrelevant features the proposed kernels are not able to outperform baseline methods. However, if there are many relevant features, and these features are related, the kernels perform significantly better.

3.1 Wavelet Analysis and Wavelet Kernels

The wavelet transform is a synthesis of ideas that emerged over many years from different fields, such as mathematics and signal processing. Generally speaking, the wavelet transform is a tool that divides up data, functions, or operators into different frequency components and then studies each component with a resolution matched to its scale (Daubechies, 1992). Therefore, the wavelet transform is anticipated to provide economical and informative mathematical representation of many objects of interest (Abramovich, Bailey, and Sapatinas, 2000). Wavelets have been widely applied in such computer science research areas as image processing, computer vision, network management, and machine learning.

Wavelet theory could naturally play an important role in machine learning since it is well founded and of very practical use. Wavelets have many favorable properties, such as vanishing moments, hierarchical and multiresolution decomposition structure, linear time and space complexity of the transformations, decorrelated coefficients, and a wide variety of basis functions (Li et al., 2002). Wavelets can provide representations of data that make the learning process more efficient and accurate (Section 3.1.4). Standard wavelet applications are mainly on data which have temporal/spatial localities (e.g. time series, stream data, and image data) wavelets have also been successfully applied to diverse domains in machine

learning (Li et al., 2002).

Compact support guarantees the localization of wavelets (in other words, processing a region of data with wavelets does not affect the the data out of this region as in Fourier transform (Section 3.1.1 and 3.1.2). A function is compactly supported if the subset of its domain where the function is nonzero is a compact set. Vanishing moment guarantees wavelet processing can distinguish the essential information from non-essential information; and dilating relation leads to fast wavelet algorithms. It is the requirements of localization, hierarchical representation and manipulation, feature selection, and efficiency in many tasks in machine learning that make wavelets a very powerful tool. The other properties such as smoothness and generators of orthonormal basis are sometimes preferred. For example, Haar wavelet is the simplest wavelet which is discontinuous, while all other Daubechies wavelets are continuous. Furthermore spline wavelets generate unconditional basis rather than orthonormal basis (Chui and Lian, 1996), and some wavelets can only generate redundant frames rather than a basis (Ron and Shen, 1995; Daubechies et al., 2003). The question whether one should use an orthonormal basis or other (such as an unconditional basis or a frame) in certain applications is yet to be answered for the general case. However, Section 3.2 argues that nonorthonormal wavelets work better as support vector kernels.

While it is convenient to apply wavelets to practical applications if one thinks of wavelets as convolution filters. However, thinking of wavelets as functions which own some special properties such as compact support, vanishing moments or multiscaling, and making use of some simple concepts of function spaces L_2 (such as nonorthonormal basis, subspace and inner product, etc.) may allow a better understanding of the basic properties of wavelets can be successfully applied in machine learning.

In signal processing fields, people usually think of wavelets to be convolution

filters which have some specially properties such as quadrature mirror filters or high pass filters (Mallat, 1989). While it is convenient to apply wavelets to practical applications if one thinks of wavelets as convolution filters. However, thinking of wavelets as functions which own some special properties such as compact support, vanishing moments or multiscaling, and making use of some simple concepts of function spaces L_2 (such as nonorthonormal basis, subspace and inner product, etc.) may allow a better understanding of the basic properties of wavelets can be successfully applied in machine learning. Thus in this dissertation wavelets are treated as functions.

3.1.1 Fourier Transform

Fourier transforms are designed for stationary signals because they are expanded as sine and cosine waves which extend in time forever, if the representation has a certain frequency content at one time, it will have the same content for all time. Hence Fourier transform is not suitable for non-stationary signals where the signal has time varying frequency.

Fourier transform is a certain linear operator that maps functions to other functions. Fourier transform decomposes a function into a continuous spectrum of its frequency components, and the inverse transform synthesizes a function from its spectrum of frequency components. For the sake of consistency with the next section, the linear operator is indicated by only a hat over the function on which the operator acts. For an $f \in L_1(\mathbb{R})$, define its Fourier transform $\hat{f}$ by

$$\hat{f}(\omega) = \int_{\mathbb{R}} f(t) \exp(-\imath\omega t) dt. \tag{3.1}$$

The Fourier transform $\hat{f}$ is bounded and uniformly continuous (Weaver, 1988).

Schwartz space is a subspace of $L_1(\mathbb{R}^n)$, and the Schwartz functions are central to Fourier theory. A function f is a Schwartz function if for each $m \geq 0, l \geq$

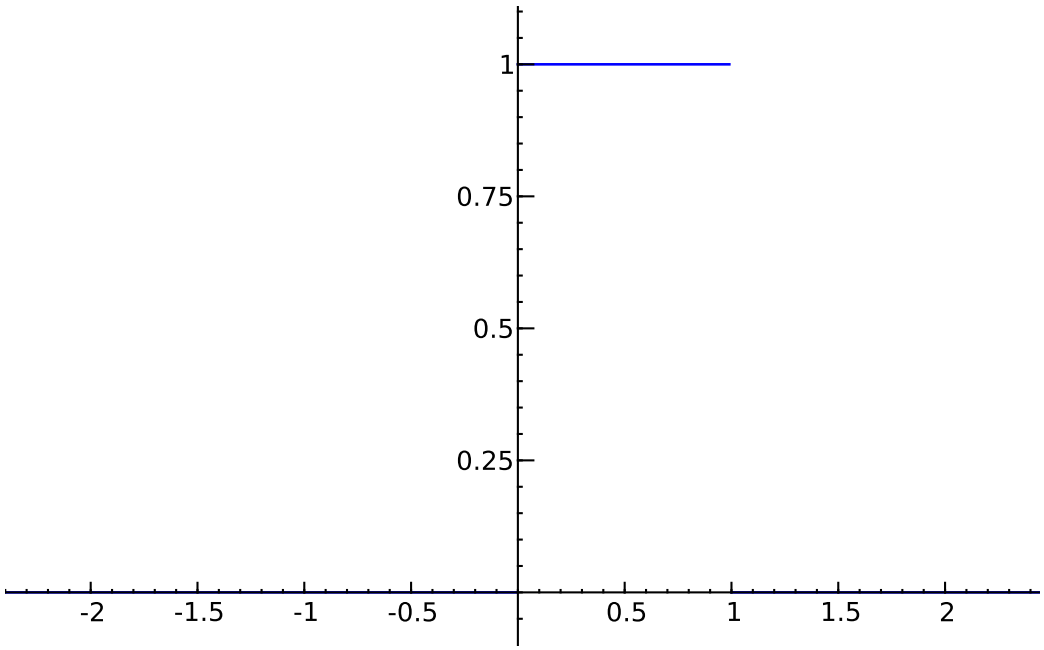


Figure 3.1: The step function is a compactly supported Lebesgue integrable function with two discontinuities.

0,

$$x^m \frac{d^l f}{dx^l}$$

is bounded on $\mathbb{R}$. Given that f is a Schwartz function, its inverse Fourier transform restores f: $f \mapsto \hat{f}$ is a one-to-one mapping from the Schwartz space S to S, and the inverse transform is defined as

$$f(t) = \frac{1}{2\pi} \int_{\mathbb{R}} \hat{f}(\omega) \exp(\imath\omega t) d\omega. \tag{3.2}$$

In signal processing $\hat{f}(\omega)$ is often called the spectrum of f , $|\hat{f}(\omega)|$ is the energy spectrum and $\arg(\hat{f}(\omega))$ is the phase (Ruskai et al., 1992).

Let f and g be integrable, and let $\hat{f}$ and $\hat{g}$ be their Fourier transforms. If f

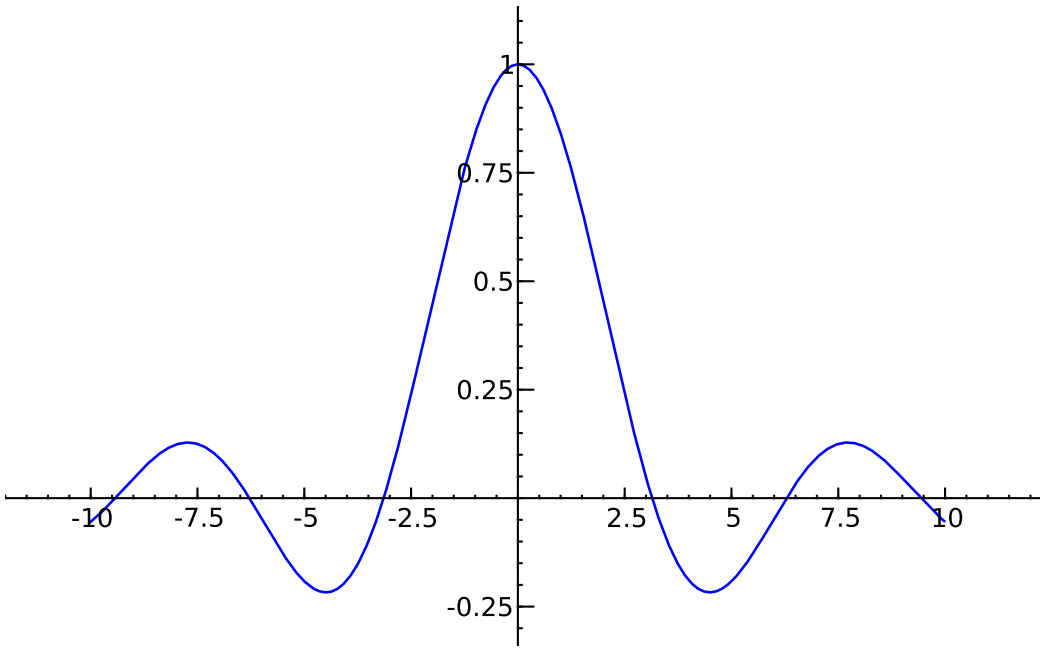


Figure 3.2: The Fourier transform of the step function is the sinc function. It is bounded and continuous, but not compactly supported and not Lebesgue integrable.

and g are also square-integrable, then we have Parseval's theorem (Rudin, 1987):

$$\int_{\mathbb{R}} f(t)\overline{g(t)}\,dt = \int_{\mathbb{R}} \hat{f}(\omega)\overline{\hat{g}(\omega)}\,d\omega, \tag{3.3}$$

where the bar denotes complex conjugation. That is, the inner product and also the norm are preserved.

Since the calculation of $\hat{f}$ requires the knowledge of f over $\mathbb{R}$, the transformation does not respect causality in signal processing. A "progressive" calculation of the transform and thus, a real-time analysis is impossible. Indeed, one cannot even approximately know the spectrum $\hat{f}$ of a signal f whose future is unknown (Ruskai et al., 1992).

Roughly speaking, the more tightly localized an f function is, the less localized its Fourier transform $\hat{f}$ is (see also Figures 3.1 and 3.2). If $f \in L_1(\mathbb{R})$, then

the duration of f is defined as follows.

$$D_0(f) = \int_{\mathbb{R}} t^2 |f(t)|^2 dt$$

The mathematical uncertainty principle states that, if f is absolutely continuous and the functions $xf(x)$ and $f'(x)$ are square integrable, then (Papoulis, 1963):

$$D_0(f) D_0(\hat{f}) \geq \frac{1}{16\pi^2} \tag{3.4}$$

This property implies that the inverse Fourier transform can be numerically unstable since useful information to reconstruct f from $\hat{f}$ with the synthesis formula may be located in the very high frequency domain. In particular this happens if f has compact support and is irregular. Fourier transform hence is an integral transformation of a global nature, and does not allow good time-frequency localization.

3.1.2 Gabor Transform

In order to overcome the disadvantage of the global nature of the Fourier transform, an idea consists of localizing the analysis by selecting a portion of the signal around a time position, conducting the Fourier analysis and then starting again for all the possible positions, enabling non-stationary signal analysis. It is the principle of the sliding window Fourier transform, also called the Gabor transform (Allen and Rabiner, 1977; Ruskai et al., 1992).

In Gabor transform the signal is divided into small segments where the signal on each of these segments could be assumed as stationary. Take a window $w \in L_1 \cap L_2$ centered in 0, with $\hat{w}$ being even and of energy 1, used to localize the analysis in time. Note that $w_{\omega,b}(t) = w(t-b)\exp(2\imath\pi\omega t)$ for $\omega, t, b \in \mathbb{R}$. The continuous Gabor transform of a signal f is defined by the following formula:

$$Gf(\omega, b) = \int_{\mathbb{R}} f(t)\overline{w_{\omega,b}(t)}dt \quad \omega, b \in \mathbb{R}.$$

As for the Fourier transform, the Gabor transform is linear, bijective on Schwarz spaces, continuous and preserves the inner products and norms. The synthesis formula is:

$$f(t) = \int_{\mathbb{R}^2} (Gf)(\omega, b) w_{\omega,b}(t) d\omega db.$$

The Gabor transform is a Fourier transform local in time, since for each value of b the Fourier transform of $f(t)\overline{w(t-b)}$ is calculated:

$$Gf(\omega, b) = \int_{\mathbb{R}} f(t)\overline{w_{\omega,b}(t)} dx = \widehat{f(t)\overline{w(t-b)}}(\omega).$$

The window w thus restrains the analysis to a domain around the position b.

Although the Gabor transform could provide a time-frequency representation of the signal, the mathematical uncertainty principle makes the choice of the segment length a big problem for it. They are complex exponentials, as for the Fourier transform, but attenuated by the window w positioned in b. The latter is zero or essentially zero (i.e. very quickly decreasing) apart from an interval centered in 0. It localizes the analyzed function in this interval (Ruskai et al., 1992). For the Gaussian window, Figure 3.3 represents three atoms defined by:

$$w_{\omega,b}(t) = \exp(-\pi(t-b)^2) \exp(2\imath\pi\omega t).$$

The number of oscillations increases with the frequency ω but the envelope is rigid and therefore the temporal resolution remains fixed. It is the major defect of this transform.

3.1.3 Wavelet Transform

Wavelet transform is designed to give good time resolution and poor frequency resolution at high frequencies and good frequency resolution and poor time resolution at low frequencies (Polikar, 1996). This is useful for many practical signals

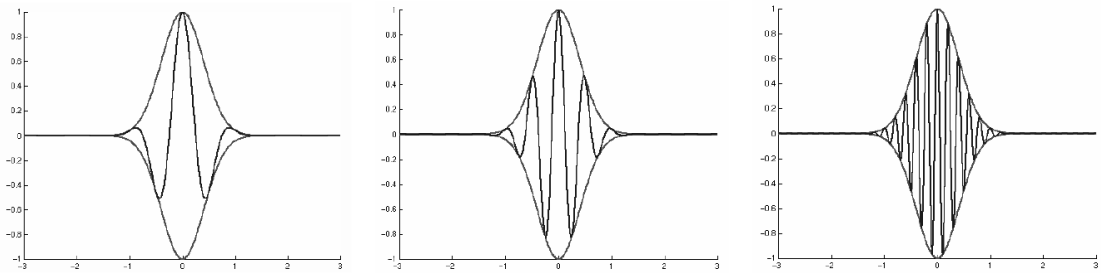


Figure 3.3: Envelope ($\pm \exp(-\pi t^2)$) and real part of the window functions for ω =1, 2 and 5. Figure adopted from (Ruskai et al., 1992).

since they usually have high frequency components for a short durations (bursts) and low frequency components for long durations (trends). The time-frequency cell structures for Gabor transform and wavelet transform are shown in Figure 3.4 and Figure 3.5 respectively (Li et al., 2002).

Wavelet analysis expresses or approximates a signal or function by a family of functions generated by dilations and translations of a function, starting with a function $\phi(t)$ that is made up of smaller version of itself. This is the refinement (or 2-scale,dilation) equation:

$$\phi(t) = \sum_{k=-\infty}^{\infty} a_k \phi(2t - k), \tag{3.5}$$

where a_ks are called filter coefficients or masks. The function $\phi(t)$ is called the scaling function (or father wavelet). The following $\psi(t)$ is called the mother wavelet (Li et al., 2002):

$$\psi(t) = \sum_{k=-\infty}^{\infty} (-1)^k \overline{a_{1-k}} \phi(2t - k).$$

From the mother wavelet, the child wavelets are derived as:

$$\psi_{j,k}(t) = |j|^{-1/2} \psi(\frac{t-k}{j}) \quad \text{over the integers } j, k.$$

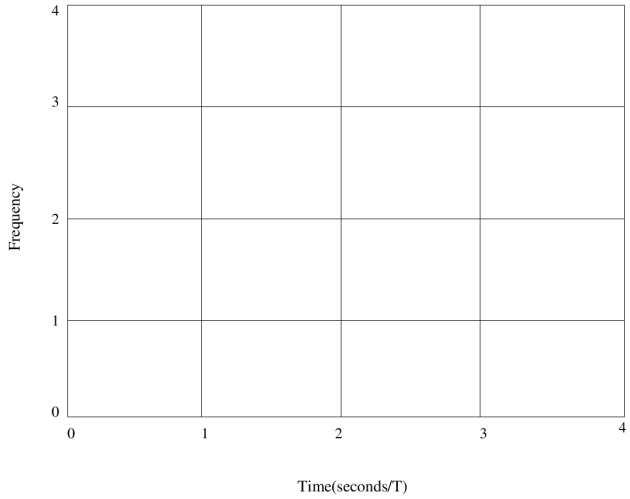


Figure 3.4: Time-frequency structure of Gabor transform. The graph shows that time and frequency localizations are independent. The cells are always square.

The child wavelets form a basis for functions in L_2, and the wavelet transform maps a function $f(t)$ onto this basis, yielding the wavelet coefficients $d_{j,k}$:

$$d_{j,k} = \int_{-\infty}^{\infty} \psi_{j,k}(t) f(t) dt.$$

A mother wavelet $\psi \in L_1 \cap L_2$ is called admissible if

$$\int_0^{\infty} \frac{|\hat{\psi}(\omega)|^2}{|\omega|} d\omega = \int_{-\infty}^{0} \frac{|\hat{\psi}(\omega)|^2}{|\omega|} d\omega < \infty.$$

Let K_ψ denote the common value of the integrals. Then the inner product is preserved (Ruskai et al., 1992):

$$(f, g)_{L_2} = \frac{1}{K_\psi} \int_{]0,\infty[\times\mathbb{R}} d_{j,k}(f) \overline{d_{j,k}(g)} \frac{djdk}{j^2}. \tag{3.6}$$

This result resembles Parseval's theorem from Fourier analysis (see Equation 3.3).

The time and frequency resolution problems are results of a theorem (the mathematical uncertainty principle) and exist regardless of the transform used.

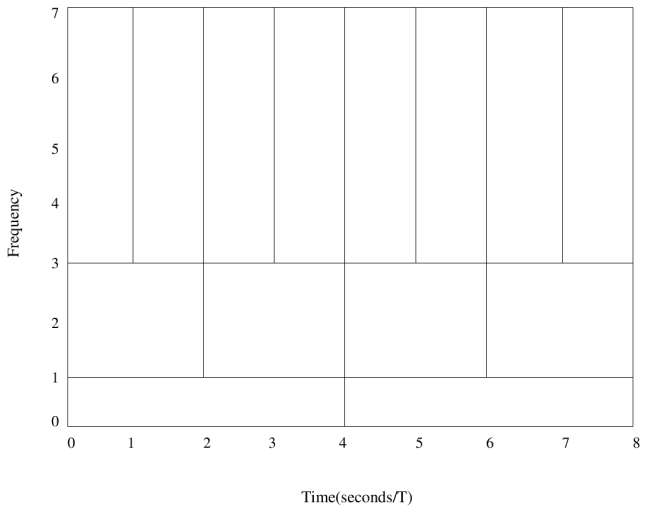


Figure 3.5: Time-frequency structure of wavelet transformation. The graph shows that frequency resolutions good for low frequency and time resolution is good at high frequencies.

Wavelet transform offers a different way to analyze any signal by using an alternative approach called the multiresolution analysis (MRA). MRA, as implied by its name, analyzes the signal at different frequencies with different resolutions. Every spectral component is not resolved equally as was the case in the Gabor transform. MRA was first introduced in (Lemarie and Meyer, 1986) and there is a fast family of algorithms based on it (Mallat, 1989). The motivation of MRA is to use a sequence of embedded subspaces to approximate $L_2(\mathbb{R})$ so that one can choose a proper subspace for a specific application task to get a balance between accuracy and efficiency (that is, larger subspaces can contribute better accuracy but waste computing resources). Mathematically, MRA studies the property of a sequence of closed subspaces $V_j, j \in \mathbb{Z}$ which approximate $L_2(\mathbb{R})$ and satisfy

$$\ldots V - -2 \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \ldots,$$

where

1. $\overline{\bigcup_{j\in\mathbb{Z}} V_j} = L_2(\mathbb{R})$ ($L_2(\mathbb{R})$ space is the closure of the union of all V_j);

2. $\cap_{j\in\mathbb{Z}} V_j = \emptyset$ (the intersection of all V_j is empty).

.

The multiresolution is reflected by the additional requirement

$$f \in V_j \Leftrightarrow f(2t) \in V_{j+1} \quad j \in \mathbb{Z}$$

This is equivalent to $f(t) \in V_0 \Leftrightarrow f(2^j t) \in V_j$, that is, all the spaces are scaled versions of the central(reference) space V_0.

The scaling function of the wavelet transform easily generates a sequence of subspaces which can provide a simple multiresolution analysis. First, the translations of $\phi(t)$, i.e., $\phi(t-k), k \in \mathbb{Z}$, span some subspace V_0 (in this subspace, $\phi(t-k), k \in \mathbb{Z}$ constitutes a basis). Similarly $2^{-1/2}\phi(2t-k), k \in \mathbb{Z}$ span another subspace, say V_1. It implies that ϕ falls into subspace V1 and so the translations $\phi(t-k), k \in \mathbb{Z}$ also fall into subspace V_1 . Thus V_0 is embedded into V_1 . With different dyadic, it is straightforward to obtain a sequence of embedded subspaces of $L_2(\mathbb{R})$ from only one function. It can be shown that the closure of the union of these subspaces is exactly $L_2(\mathbb{R})$ and their intersections are empty sets (Daubechies, 1992). Here j controls the observation resolution while k controls the observation location (see Section 3.2).

Given two consecutive subspaces, say V_0 and V_1 , the complement space $V_1 \backslash V_0$ is usually denoted as W_0 . The mother wavelet ψ also falls into V_1 (and so do its translations $\psi(t-k), k \in \mathbb{Z}$). Notice that ψ is orthogonal to ϕ. It is easy to claim that an arbitrary translation of the father wavelet ϕ is orthogonal to an arbitrary translation of the mother wavelet ψ. Thus, the translations of the wavelet ψ span the complement subspace W_0 . Similarly, for an arbitrary $j, \psi_{k,j}, k \in \mathbb{Z}$, span an orthonormal basis of W_j which is the orthogonal complement space of V_j in V_{j+1} .

Therefore, $L_2(\mathbb{R})$ space is decomposed into an infinite sequence of wavelet spaces, i.e., $L_2(\mathbb{R} = \bigoplus_{j\in\mathbb{Z}} W_j$.

A direct application of multiresolution analysis is the fast discrete wavelet transform algorithm, called pyramid algorithm. The core idea is to progressively smooth the data using an iterative procedure and keep the detail along the way, i.e., analyze projections of f to W_j (Mallat, 1989). This property was also used in a wavelet kernel to offer an alternative to the computationally costly singular value decomposition (see also the next section) (Hoenkamp, 2003).

The fact that $L_2(\mathbb{R})$ is decomposed into an infinite wavelet subspace is equivalent to the statement that $\psi_{j,k}, j, k \in \mathbb{Z}$ span a basis of $L_2(\mathbb{R})$. An arbitrary function $f \in L_2(\mathbb{R})$ then can be expressed as follows:

$$f(t) = \sum_{j,k\in\mathbb{Z}} d_{j,k}\psi_{j,k}(t),$$

where $d_{j,k} = (f, \psi_{j,k})$ is called wavelet coefficients. Note that j controls the observation resolution and k controls the observation location. If data in some location are relatively smooth (it can be represented by low-degree polynomials), then its corresponding wavelet coefficients will be fairly small by the vanishing moment property of wavelets.

3.1.4 Wavelet Kernels

Wavelet kernels have been introduced for both support vector regression and classification. Most of these wavelet kernels do not use the inner product of the embedding space, but use wavelets in a similar fashion to radial basis function kernels.

Following (Zhang, Zhou, and Jiao, 2004), let $\psi(t)$ be a mother wavelet and let j and k denote the dilation and translation, respectively. If $\mathbf{x}, \mathbf{y} \in \mathbb{R}^M$, then the wavelet kernels are:

$$K(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^{M} \psi(\frac{x_i - k}{j})\psi(\frac{y_i - k}{j}), \tag{3.7}$$

and the translation-invariant wavelet kernels are:

$$K(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^{M} \psi(\frac{x_i - y_i}{j}). \tag{3.8}$$

The authors focused on the latter kernel and mainly on support vector regression using the wavelet $\psi(t) = \cos(1.75t)\exp(-\frac{t^2}{2})$ adopted from (Szu, Telfer, and Kadambe, 1992). This wavelet is not compactly supported. Results confirmed earlier findings that wavelets can be very efficient in classification problems (Sheikholeslami, Chatterjee, and Zhang, 1998). Other authors proposed similar, autocorrelation kernels for pattern recognition (Chen and Bhattacharya, 2006; Chen and Xie, 2007).

Hoenkamp has identified another wavelet kernel for information retrieval (Hoenkamp, 2003). He introduced the Haar transform as an alternative to singular value decomposition (SVD) for the following reasons.

1. The operator should be unitary to preserve cohesion of objects, so the operator maps the space to a new space in which related objects stay together and unrelated objects stay unrelated.

2. The operator should lead to dimension reduction.

3. The operator should be computationally less expensive than SVD.

The Haar wavelet's mother wavelet function $\psi(t)$ can be defined as

$$\psi(t) = \begin{cases} 1 & 0 \leq t < 1/2, \\ -1 & 1/2 \leq t < 1, \\ 0 & \text{otherwise.} \end{cases}$$

and its scaling function $\phi(t)$ can be described as

$$\phi(t) = \begin{cases} 1 & 0 \leq t < 1, \\ 0 & \text{otherwise.} \end{cases}$$

Following Hoenkamp's example, the discrete version of the Haar transform relies on the following simple observation: Any two numbers can be represented by their average, while the number that you add on one side, you subtract from the other. Haar transform can be viewed as a series of averaging and differentiating operations on a discrete function. Figure 3.6 shows the construction for an object vector of four features: (2, 0, 3, 5). The numbers are taken two by two, and represented by their average and the coefficient of a Haar function. The procedure is repeated until it ends in an overall average and a series of Haar coefficients.

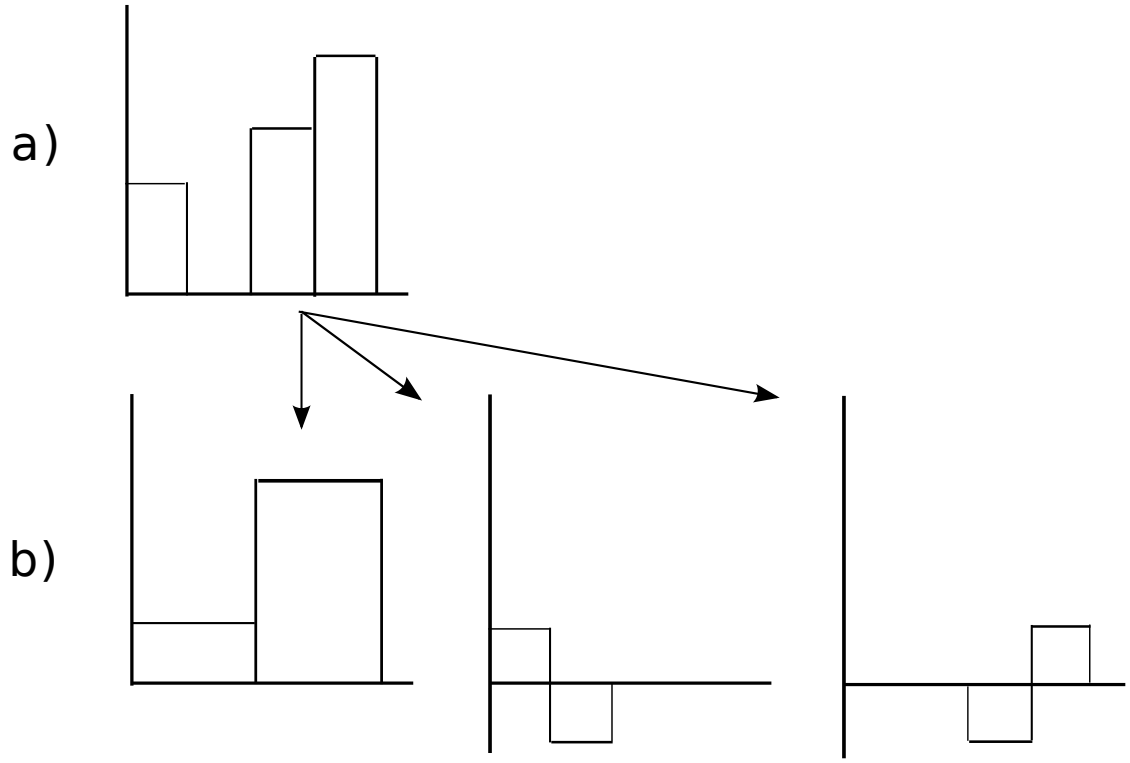


Figure 3.6: The first step of Haar expansion for an object vector (2,0,3,5). (a) the vector as a function of t. (b) Each pair of features is decomposed into its average and a suitably scaled Haar function.

The advantages of SVD is that it removes noise from the data by dimension reduction and by discarding the smallest singular values of the decomposition. The

analog for the Haar transform is to ignore the smallest Haar coefficients. If we denote the operator which maps the space to a new space in which the smallest Haar coefficients are discarded by $\hat{\phi}$, then the underlying kernel is

$$K(\mathbf{x}, \mathbf{y}) = (\hat{\phi}(\mathbf{x}), \hat{\phi}(\mathbf{y})). \tag{3.9}$$

In this regard, this wavelet kernel is different from the above two defined by Equations 3.7 and 3.8, since the inner product of the embedding space is explicitly calculated.

However, when calculating the average of a pair of features, it is assumed that there is some relation between the two features. For instance, in image processing, the relationship is spatial, and in time series analysis, the relationship is temporal. Hoenkamp introduced the Haar kernel for information retrieval, where the features are index terms of natural language documents, and relatedness is not guaranteed for two consecutive terms in the vector representation of a document. The assignment of canonical basis vectors to features is arbitrary in case of the classical vector space-based model. Let M denote the number of features. By the classical approach, one vector of the canonical basis $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_M\}$ of $\mathbb{R}^M$ is assigned to each feature. The assignment of features to vectors is arbitrary. Let x_{ij} be the weight of feature f_i in object x_j. Thus an object vector $\mathbf{x}_j$ is a linear combination of the canonical basis vectors.

$$\mathbf{x}_j = \sum_{i=1}^{M} x_{ij}\mathbf{e}_i.$$

By writing $\mathbf{x_j}$ as a column vector, $\mathbf{x}'_j = (x_{1j}, x_{2j}, \ldots, x_{Mj})$.

Since the basis vectors of the canonical basis are perpendicular to one another, it implies that the features are mutually independent, an assumption that is rarely valid. One important thing to note is that the assignment of features to vectors is completely *arbitrary*, a feature can be assigned to any of the vectors of the canonical basis.

In a similar approach, voice signals have been transformed by Daubechies wavelets, approximation coefficients have been removed, and SVMs have been trained on the inverse transformation (Fonseca et al., 2005; Fonseca et al., 2007). It is also possible to train SVMs directly on the wavelet coefficients, without using an inverse transformation (Tuntisak and Premrudeepreechacharn, 2007; Hosseini et al., 2008). In these cases, a temporal relation existed between the features. SVMs trained on the wavelet coefficients of spatial information have also been investigated (Schleif et al., 2009; Alexandrov et al., 2009).

3.2 Compactly Supported Basis Functions as Support Vector Kernels

We define compactly supported basis functions (CSBF) as functions of a basis of L_2(ℝ with a compact support, that is, they vanish outside of a compact set, in our case, a compact interval. Some wavelet functions, such as the Haar transform or B-spline wavelets, have a compact support. CSBF kernels in turn are a kind of wavelet kernel functions.

The proposed CSBF kernel heavily utilizes feature interdependence, and hence the arbitrary assignment of features to basis vectors discussed in the previous section should be avoided. Instead, related features are assigned to *subsequent* vectors of the canonical base. To this end, features should be grouped together based on a statistical distance or metric (see Section 3.5).

If we assume that a set of features is reordered according to some relatedness measure, then the vector representation of an object may be regarded as a series of equally spaced observations of a continuous signal, where the consecutive observations are not in a temporal relation with one another, but the relation is defined by statistical relatedness.

We can consider reconstructing the hypothetical signal. The Whittaker-Shannon formula gives a simple example of how to reconstruct the signal from discrete values (Weaver, 1988). If we denote the reconstructed signal by $\hat{s}_{\mathbf{x}}$ of a vector $\mathbf{x}$, the Whittaker-Shannon formula is defined by

$$\hat{s}_{\mathbf{x}}(t) = \sum_{k=0}^{n-1} x_k \text{sinc}(2\pi\omega(t-k)), \quad t \in [1, M]. \tag{3.10}$$

where t is a dummy variable, $\text{sinc(t)} = \frac{\sin(t)}{t}$, and ω reflects prior knowledge of the sampling, and affects the width of the sinc function. More generally, the formula is

$$\hat{s}_{\mathbf{x}}(t) = \sum_{k=0}^{n-1} x_k b(t-k), \quad t \in [1, M], \tag{3.11}$$

with an appropriately chosen basis function $b(t)$ of L_2, or a subspace thereof.

The inner product of the $L_2[1, M]$ space is applied to express similarity between objects. Let λ be the Lebesgue measure on $\mathbb{R}$, then the inner product is defined as:

$$(f, g) = \int_{[1,M]} f(t)g(t)d\lambda(t), \quad f, g \in L_2([1, M]). \tag{3.12}$$

Using Equation 3.11, the CSBF kernel is defined as:

$$K(\mathbf{x}, \mathbf{y}) = (\hat{s}_{\mathbf{x}}, \hat{s}_{\mathbf{y}}) = \int_{[1,M]} \sum_{k=1}^{M} \sum_{l=1}^{M} x_k y_l b(t-k)b(t-l)d\lambda(x) = \tag{3.13}$$

$$\sum_{k=1}^{M} \sum_{l=1}^{M} x_k y_l \int_{[1,M]} b(t-k)b(t-l)d\lambda(t).$$

The term $\int_{[1,M]} b(t-k)b(t-l)d\lambda(t)$ is an overhead in calculating the similarity, hence w should be chosen in a way that minimizes this overhead.

A matching feature in two objects will be counted to its full score as in the vector space, while nearby related features will be counted less and less according their proximity to the matching feature. Assuming that the related features f_{i-1}, f_i, and f_{i+1} follow each other, consider the following example. The first object has

the feature f_i, and so does the second object. In Figure 3.7, it can be seen that feature f_i is counted the same way as it would be in a vector space model, the related features f_{i-1} and f_{i+1} are counted to a smaller extent, while other related features are considered even less.

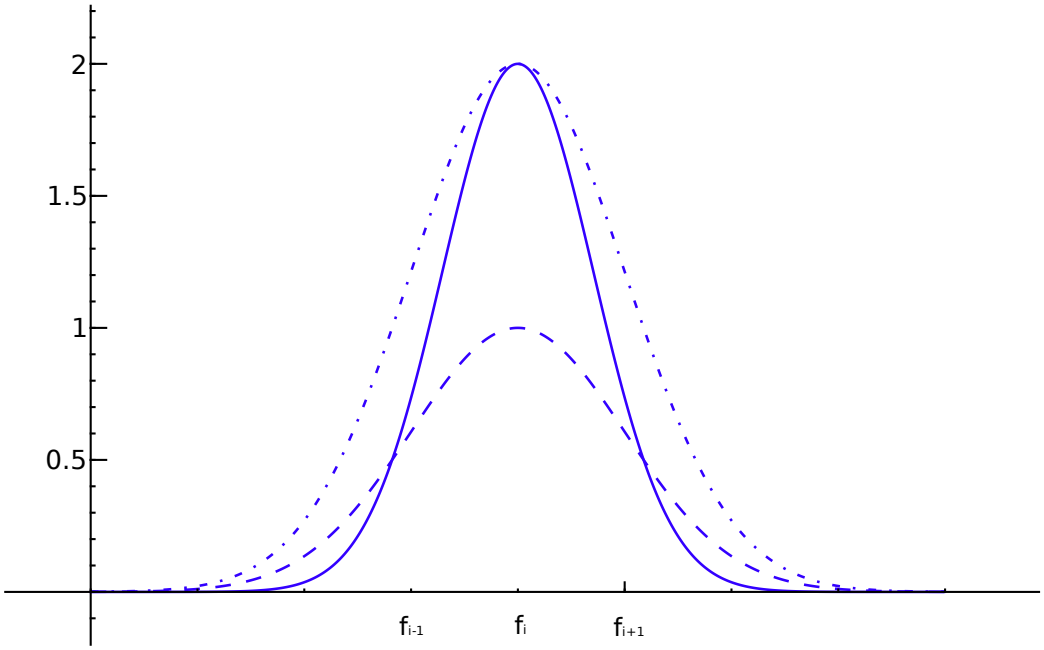


Figure 3.7: Two objects with a matching feature f_i. Dotted line: Object-1. Dashed line: Object-2. Solid line: Their product as in Equation (3.12).

Now if the two objects do not share the exact feature, only related features occur, for instance, f_{i-1} and f_{i+1}, respectively, then the feature f_i, placed between f_{i-1} and f_{i+1} in the same order, will be considered to some extent for the calculation of similarity (see Figure 3.8).

To calculate the kernel, let us expand Equation 3.13:

$$K(\mathbf{x}, \mathbf{y}) = (\hat{s}_{\mathbf{x}}, \hat{s}_{\mathbf{y}})_{L_2} = \tag{3.14}$$

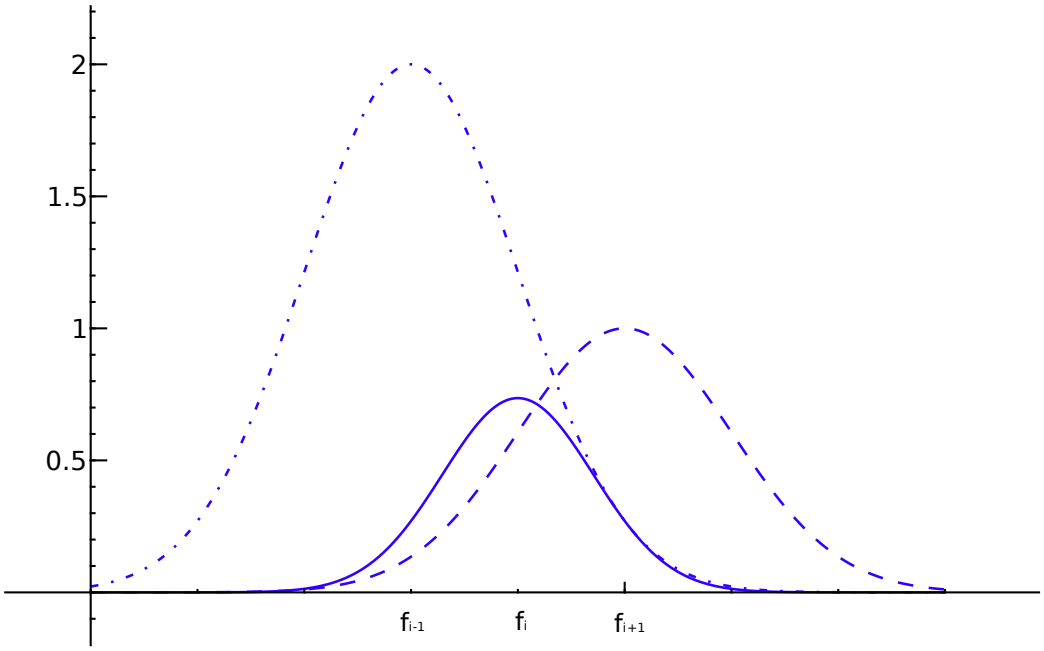


Figure 3.8: Two objects with no matching features but with related features f_{i-1} and f_{i+1}. Dotted line: Object-1. Dashed line: Object-2. Solid line: Their product as in Equation (3.12).

$$\sum_{k=1}^{M}\sum_{l=1}^{M} x_k y_l \int_{[1,M]} b(t-k)b(t-l)d\lambda(t).$$

To simplify the above sum, we suggest to use non-orthogonal basis functions with a compact support. An orthogonal basis would give zero for all value of k and l safe for $k = l$, while basis functions with a non-compact support would require M^2 calculations for each $\mathbf{x}_i$ and $\mathbf{x}_j$. If the support of the function is a continuous compact interval of length $2b$, then

$$K(\mathbf{x},\mathbf{y}) = (\hat{s}_{\mathbf{x}}, \hat{s}_{\mathbf{y}})_{L_2} = \tag{3.15}$$

$$\sum_{k=1}^{M}\sum_{l=\max\{k-\lceil b\rceil,1\}}^{\min\{j+\lceil b\rceil,M\}} x_k y_l \int_{[1,M]} b(t-k)b(t-l)d\lambda(t).$$

Thus the eventual kernel evaluation cost is O(bM).

The choice of the width may be more important than the actual functional form of the kernel. There may be little difference in the relevant part of the filter properties between e.g. a B-Spline and a Gaussian kernel (Smola, Schölkopf, and Müller, 1998). Therefore we focus on a kernel which is easy to compute, and study the impact of width, instead of the choice of function. Choosing a small width of the kernels leads to high generalization error as it effectively decouples separate basis functions of the kernel expansion into very localized functions which is equivalent to memorizing the data, while a wide kernel tends to oversmooth (Smola, Schölkopf, and Müller, 1998).

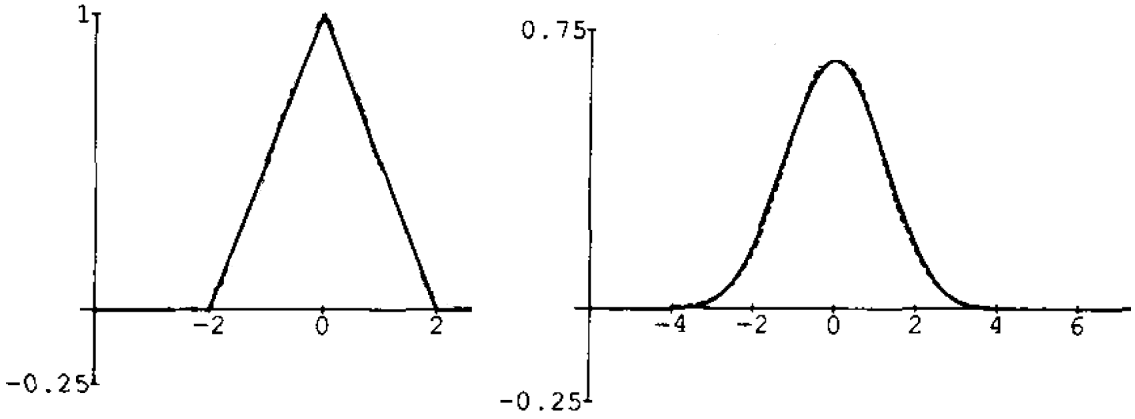


Figure 3.9: First and third order B-splines. Figure adopted from (Unser et al., 1992).

Spline wavelets are extremely regular and usually symmetric or anti-symmetric. They can be designed to have a compact support (Unser, 1997). A B_n-spline is defined as

$$B_n(t) = \sum_{r=0}^{n+1} \frac{(-1)^r}{n!} \binom{n+1}{r} \left(t + \frac{n+1}{2} - r \right)_+^n ,$$

where $(.)_+ = \max\{., 0\}$. B_n has a compact support $[-\frac{n+1}{2}, \frac{n+1}{2}]$.

Focusing on the integration part of Equation 3.15, by using the convolution property of B_n splines (Unser and Aldroubi, 1993), we get

$$\int_{\mathrm{R}} B_n(t-k)B_n(l-t)d\lambda(t) = B_{2n+1}(k-l).$$

The above formula makes computations easier.

In the above, we assumed that the vector is a sequence of equally spaced observations. However, the observations do not have to be equally spaced, they can be randomly spaced by employing a scaling function $s(x)$ on the indices of features:

$$s(k) = \sum_{i=1, i<k} d(f_i, f_{i+1}).$$

The scaling function allows that features that are closer to each other get a higher score when calculating the kernel:

$$K(\mathbf{x}, \mathbf{y}) = (\hat{s}_{\mathbf{x}}, \hat{s}_{\mathbf{y}})_{L_2} = \tag{3.16}$$
$$\sum_{k=1}^{M}\sum_{l=1}^{M} x_k y_l \int_{[1,M]} b(t-s(k))b(t-s(l))d\lambda(t).$$

3.3 Validity of CSBF Kernels

The concept of kernel in machine learning comes from linear analysis, in particular from the theory of integral operators. Mercer's theorem cleared the way to nonlinear classifiers.

Let X be a subset of $\mathbb{R}^n$, and K be a complex valued Lebesgue measurable function on $X \times X$ such that

$$\int_X \int_X |K(\mathbf{t}, \mathbf{s})|^2 d\lambda(\mathbf{s}) d\lambda(\mathbf{t}) < \infty,$$

where λ is the Lebesgue measure on $\mathbb{R}^n$ (Gohberg and Goldberg, 1980). The function K is the kernel of the integral operator $T_K : L_2(X) \to L_2(X)$

$$(T_K f)(\mathbf{t}) = \int_X K(\mathbf{t}, \mathbf{s}) f(\mathbf{s}) d\lambda(\mathbf{s}).$$

Theorem.(Mercer) Let X be a compact subset of $\mathbb{R}^n$ and $f \in L_2(X)$. Suppose K is a continuous symmetric kernel on X such that the integral operator $T_K : \mathcal{L}_2(X) \to \mathcal{L}_2(X)$,

$$(T_K f)(\mathbf{t}) = \int_X K(\mathbf{t}, \mathbf{s}) f(\mathbf{s}) d\lambda(\mathbf{s}),$$

is positive, that is

$$\int_{X \times X} K(\mathbf{x}, \mathbf{z}) f(\mathbf{x}) f(\mathbf{z}) d\lambda(\mathbf{x}) d\lambda(\mathbf{z}) \geq 0,$$

for all $f \in L_2(X)$. Then $K(\mathbf{x}, \mathbf{z})$ can be expanded in a uniformly convergent series (on $X \times X$) in terms of T_K's eigenfunctions $\phi_j \in L_2(X)$, normalized in such a way that $\|\|\phi_j\|\|_{L_2} = 1$, and nonnegative associated eigenvalues $\lambda_j \geq 0$,

$$K(\mathbf{x}, \mathbf{z}) = \sum_{j=1}^{\infty} \lambda_j \phi_j(\mathbf{x}) \phi_j(\mathbf{z}). \tag{3.17}$$

(See (Gohberg and Goldberg, 1980) for a proof).

Equation (3.17) is a generalization of the inner product in a Hilbert space introducing a weighting λ_j for each dimension. This property lets support vector machines replace the inner product by a nonlinear kernel which satisfies the conditions of Mercer's theorem.

Let K_1 and K_2 be kernels over $X \times X$, $X \subset \mathbb{R}^n$, $a \in \mathbb{R}^+$, f a real-valued function on X and in $L_2(X)$, K_3 a kernel over $Y \times Y$, $Y \subset \mathbb{R}^m$, $\theta : X \to Y$, and $\mathbf{B}$ a symmetric positive semi-definite $n \times n$ matrix. For all $x, z \in X$, $x', z' \in Y$, and $a \in \mathbb{R}$, the following functions are also kernels (Cristianini and Shawe-Taylor, 2000):

1. $K(\mathbf{x}, \mathbf{z}) := K_1(\mathbf{x}, \mathbf{z}) + K_2(\mathbf{x}, \mathbf{z})$,

2. $K(\mathbf{x}, \mathbf{z}) := aK_1(\mathbf{x}, \mathbf{z})$,

3. $K(\mathbf{x}, \mathbf{z}) := K_1(\mathbf{x}, \mathbf{z}) K_2(\mathbf{x}, \mathbf{z})$,

4. $K(\mathbf{x}, \mathbf{z}) := f(\mathbf{x})f(\mathbf{z})$,

5. $K(\mathbf{x}, \mathbf{z}) := K_3(\theta(\mathbf{x}), \theta(\mathbf{z}))$,

6. $K(\mathbf{x}, \mathbf{z}) := \mathbf{x}'\mathbf{B}\mathbf{z}$.

The motivation for developing the CSBF kernel was to overcome the flaws of the vector space model, and provide a more intuitive way of knowledge representation. However, the developed model can be embedded easily into the existing machine learning methods. The similarity measure of the CSBF model defined by Equation 3.12 is an inner product. An inner product is a nondegenerate sesquilinear form, moreover it is positive definite. Let ϕ be the mapping that creates the L_2 functions from the discrete data, and X be a subspace of $\mathbb{R}^M$:

$$\phi : X \rightarrow L_2([1, M]).$$

The kernel underlying the model is

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\phi(\mathbf{x}_i), \phi(\mathbf{x}_j))_{L_2}. \tag{3.18}$$

This kernel is positive definite, therefore the conditions of Mercer's Theorem are satisfied. It implies that the kernel defined by Equation 3.18 can be used by support vector machines (Section 2.2.6).

3.4 Computational Complexity of CSBF Kernels

The proposed kernel does not change the computational complexity of support vector machines other than the kernel evaluation cost. Given a linear kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i, \mathbf{x}_j),$$

the computational cost of a kernel evaluation is $\mathrm{O}(M)$, if M is the number of features:

$$(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k}^{M} x_{ki} x_{kj}.$$

To calculate the evaluation cost for the CSBF kernels, let us start from Equation 3.13:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\phi(\mathbf{x}_i), \phi(\mathbf{x}_j))_{L_2} = \sum_{k=1}^{M} \sum_{l=1}^{M} x_{ki} x_{lj} \int_{[1,M]} b(t-k) b(t-l) d\lambda(t). \quad (3.19)$$

The above formula indicates a magnitude increase in the complexity ($\mathrm{O}(M^2)$), however, it can be simplified. Since the approximating function $b(x)$ has a compact support b, it is reasonable to calculate the score only for data points within the support, that is, for a feature j, the interval $[\max\{j - \lceil b \rceil, 1\}, \min\{j + \lceil b \rceil, M\}]$. Hence the above equation simplifies to

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\phi(\mathbf{x}_i), \phi(\mathbf{x}_j))_{L_2} = \sum_{k=1}^{M} \sum_{l=\max\{k-\lceil b \rceil, 1\}}^{\min\{j+\lceil b \rceil, M\}} x_{ki} x_{lj} \int_{[1,M]} b(t-k) b(t-l) d\lambda(t). \quad (3.20)$$

Thus the eventual kernel evaluation cost is $\mathrm{O}(bM)$.

3.5 An Algorithm to Reorder the Feature Set

As mentioned in Section 3.2, the proposed CSBF kernels heavily utilize feature interdependence, therefore features that are related to each other should be adjacent when they are assigned to the vectors of the canonical basis. An ordering for the CSBF kernel should address the following issues.

1. The ordering is possible. For example, consider features, assume that they are in order and have their positions i and j respectively. This assumption implicitly suggests that one can jump from i to j alongside a continuum where the first features gradually shifts into the second one.

2. The ordering is good enough. Features that are completely unrelated will not be adjacent in the ordering.

A k nearest neighbor clustering algorithm was tested for the ordering of terms with the Lesk dissimilarity as the distance function (Wittek and Darányi, 2007) in a text classification scenario. However, it is evident that the first of the above assumptions is tenable within a cluster, but it is equally evident that it is not tenable between the clusters: given the terms *dog* and *ship*, a *dog*-related cluster can be adjacent to a *ship*-related. This section proposes an algorithm which creates a feature ordering directly using a heuristic to find a minimum-weight Hamiltonian path, and thus will be referred to as ordering based on Hamiltonian path (OHP).

Let V denote a set of features $\{f_1, f_2, \ldots, f_n\}$ and let $d(f_i, f_j)$ denote the distance between the features f_i and f_j. The initial order of the features is not relevant.

Let $G = (V, E)$ denote a weighted undirected graph, where the weights on the set E are defined by the distances between the features (Figure 3.10). G is a K_n complete graph. This graph is similar to the one used in (Davidson, Wylie, and Boyack, 2001) for 3D visualization of text similarity, but this graph is complete to enable a full exploitation of similarity relations.

Finding an ordering of features can be translated to a graph problem: a minimum-weight Hamiltonian path G' of G gives the ordering by reading the nodes from one end of the paths to the other. For instance, given three features with distance $d(f_1, f_2) = 2$, $d(f_1, f_3) = 1$, and $d(f_2, f_3) = 3$, the minimum-weight Hamiltonian path will be $f_3 \rightarrow f_1 \rightarrow f_2$ (Figure 3.11), and the respective feature order will be $f_3,\ f_1,\ f_2$.

G is a complete graph, therefore such a path always exists, but finding it is an NP-complete problem. The following greedy algorithm is similar to the nearest neighbor heuristic for the solution of the traveling salesman problem. It creates a graph $G' = (V', E')$, where $V' = V$ and $E' \subset E$. This G' graph is a spanning tree of G in which the maximum degree of a node is two, that is, the minimum spanning

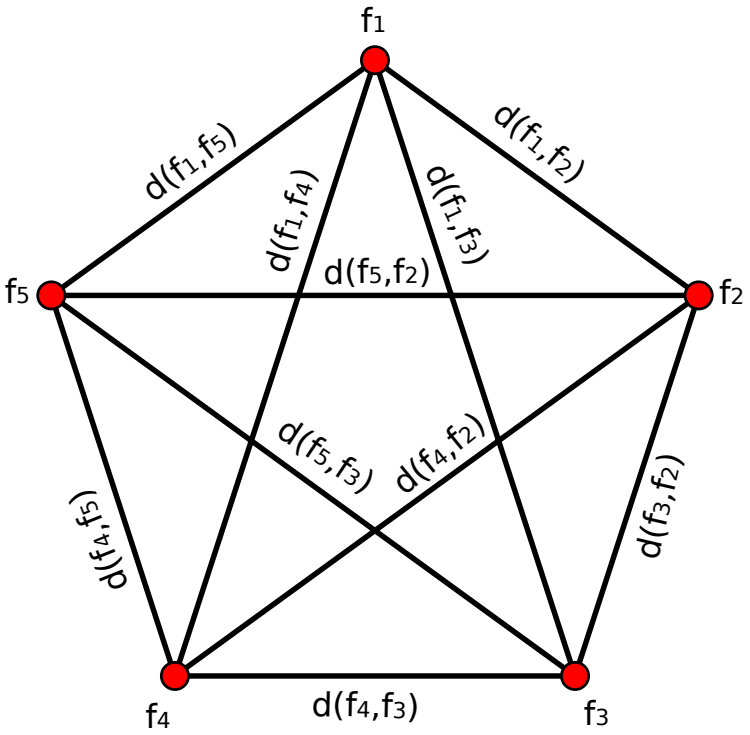


Figure 3.10: A weighted K_5 for a feature set of five elements.

tree is a path between two nodes.

Step 1 Find a seed feature f_s. If an external knowledge base does not exist to give clues to finding the seed feature, choose it randomly. This seed feature is the first element of V', $V' = \{f_s\}$. Remove it from the set V:

$$V := V \backslash \{f_s\}.$$

Step 2 Let f_l denote the leftmost feature of the ordering and f_r the rightmost one. Find the next two elements of the ordering:

$$f'_l = \text{argmin}_{f_i \in V} d(f_i, f_l),$$

$$f'_r = \text{argmin}_{f_i \in V \backslash \{f'_l\}} d(f_i, f_r).$$

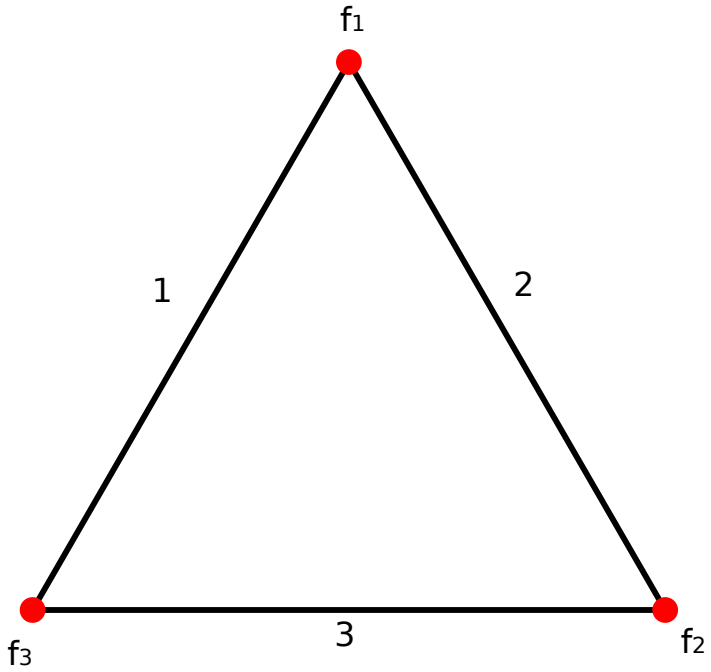


Figure 3.11: A weighted K_3 for a feature set of three elements with example weights.

Step 3 If $d(f_l, f'_l) < d(f_r, f'_r)$ then add f'_l to V', $E' := E' \cup \{e(f_l, f'_l)\}$, and $V := V \backslash \{f'_l\}$. Else add f'_r to V', $E' := E' \cup \{e(f_r, f'_r)\}$ and $V := V \backslash \{f'_r\}$ (Figure 3.12).

Step 4 Repeat from *Step 2* until $V = \emptyset$.

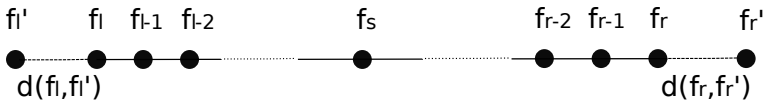


Figure 3.12: An intermediate step of the ordering algorithm

The above algorithm can be thought of as a modified Prim's algorithm, but it does not find the optimal minimum-weight spanning tree. The algorithm will always terminate, that is, the ordering is always possible, since finding a minimum distance on the remaining set takes finite steps, and the remaining set becomes strictly smaller with each iteration. However, the quality of the ordering varies (see Section 3.8.1), as the greedy heuristic may not be able to find the global optimum.

The resulting order apparently depends on the distance function. In case of a distributional distance, the actual order will depend on the training data, hence it may be skewed and may introduce additional noise if the test data is substantially different. On the other hand, if it only depends on an external knowledge source, it will be independent of the data set, and thus it is unable to reflect the qualities of the data. A composite measure appears to be the best choice, as it will be shown in the next chapter.

Without any index support, the computational cost of the ordination algorithm is $O(n^2)$, since a full scan of the set of terms is required in Step 2. If a tree-based spatial index can be used, the run-time is reduced to $O(n \log n)$ since searches are supported efficiently by spatial access methods such as the R*-tree (Beckmann et al., 1990) or the X-tree (Berchtold, Keim, and Kriegel, 2001) for data from a vector space or the M-tree (Ciaccia, Patella, and Zezula, 1997) for data from a metric space. The height of such a tree-based index is $O(logn)$ for a database of n objects in the worst case and, at least in low-dimensional spaces, a search with a "small" search region has to traverse only a limited number of paths.

The resulting order is similar to the order generated by Ordering Points To Identify the Clustering Structure (OPTICS), a modified version of density clustering (Ankerst et al., 1999).

OPTICS was derived from Density-Based Spatial Clustering of Applications with Noise (DBSCAN, (Ester et al., 1996)), which needs two input parameters, ϵ

and MinPts, to define:

1. An ϵ-neighborhood $N_\epsilon(x) = \{y \in X | d(x, y) \leq \epsilon\}$ of the point x;
2. A core object (a point with a neighborhood consisting of more than MinPts points);
3. A concept of a point y density-reachable from a core object x (a finite sequence of core objects between x and y exists such that each next belongs to an ϵ-neighborhood of its predecessor);
4. A symmetric relation density-connectivity of two points x, y (they should be density-reachable from a common core object).

All the points reachable from core objects can be factorized into maximal connected components serving as clusters. The points that are not connected to any core point can be considered as outliers, because they are not covered by any cluster. The run time complexity of DBSCAN is O$n(\log n)$. With regard to the two parameters ϵ and MinPts, there is no straightforward way to fit them to data. To overcome this obstacle, the algorithm OPTICS was developed (Ankerst et al., 1999). It builds an augmented ordering of data which is consistent with DBSCAN, but goes one step further: instead of just one point in the parameter space, OPTICS covers a spectrum of all different $\epsilon' \leq \epsilon$. The constructed ordering can be used automatically or interactively. With each point, OPTICS stores only two additional fields, the so-called core- and reachability-distances. For example, the core-distance is the distance to MinPts-nearest neighbor when it does not exceeds ϵ, or undefined otherwise. Experimentally, OPTICS exhibits runtime roughly equal to 1.6 of DBSCAN runtime, while maintaining the same complexity.

Since the time complexity of OPTICS and OHP are the same, the crucial difference between them is that the latter is non-parametric. The two methods are compared in Section 3.8.

3.6 Efficient Implementation

By a proper selection of basis functions, the overhead in the computation of Equation 3.13 can be reduced. However, it was observed that convergence to the optimal solution for the dual problem of SVM (see Formula 2.5) is a magnitude slower on average compared to the linear kernel.

The quadratic programming problem of the dual form can be written as

$$\min \frac{1}{2}\alpha' Q\alpha + \mathbf{e}'\alpha \tag{3.21}$$

subject to

$$0 \leq \alpha_i \leq C, \quad i = 1, \ldots, l$$

$$\mathbf{y}'\alpha = 0,$$

where $\mathbf{e}$ is the vector of all ones, Q is an n by n positive semi-definite matrix, $Q_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$. If the data set has extremely sparse vectors, one may not need to decompose the problem to smaller subproblems, but methods such as the cutting-plane algorithm may be applied on the primary problem (Joachims, 1999).

Solving Problem 3.21 is difficult because Q_{ij} is in general not zero, Q is a fully dense matrix, so it would need a prohibitive amount of memory to store the matrix (Hsu and Lin, 2002b). Therefore traditional optimization algorithms such as Newton, Quasi Newton, etc., cannot be applied. Several algorithms separate the training data in to two sets B and N, where B is the working set and N contains the indices not in the working set (Osuna, Freund, and Girosi, 1997; Joachims, 1999; Hsu and Lin, 2002b). By denoting α_B and α_N as vectors containing the respective elements, and noting that α_N is fixed in an iteration, Problem 3.21 can be rewritten as:

$$\min \frac{1}{2}\alpha'_B Q_{BB}\alpha_B + (\mathbf{e}'_B - Q_{BN}\alpha_N)\alpha_B \tag{3.22}$$

subject to

$$0 \leq (\alpha_B)_i \leq C, \quad i = 1, \ldots, q$$

$$\mathbf{y}'_B \alpha_B = -\mathbf{y}'_N \alpha_N,$$

where $Q = \begin{bmatrix} Q_{BB} & Q_{BN} \\ Q_{NB} & Q_{NN} \end{bmatrix}$, $Q'_{BN} = Q_{NB}$, is a permutation of the matrix Q and q is the size of the working set. Several issues emerge with such decompositions (Joachims, 1999):

1. An efficient and effective method for selecting the working set. An extreme case is the Sequential Minimal Optimization (SMO) (Platt, 1999), which restricts B to have only two elements. Then in each iteration one solves a simple two-variable problem without needing optimization software.

2. Successive shrinking of the optimization problem. This exploits the property that many SVM learning problems have

   - much less support vectors than training examples.
   - many support vectors which have an i at the upper bound C.

   The shrinking technique reduces the size of the working problem without considering some bounded variables. Near the end of the iterative process, the decomposition method identifies a possible set A where all final free α_i may reside in. If the number of iterations is large, then shrinking can shorten the training time (Chang and Lin, 2001).

3. Computational improvements like caching. An effective technique for reducing the computational time is caching. Since Q is fully dense and may not be stored in the computer memory, elements Q_{ij} are calculated as needed. One can use a special storage called cache to store recently used Q_{ij}. Then some kernel elements do not need to be recalculated. Most applications use a least-recent-use strategy for caching (Joachims, 1999; Chang and Lin, 2001).

	True	False
Predicted Positive Value	True Positive	False Positive
Predicted Negative Value	False Negative	True Negative

Table 3.1: Classification of predictions by a binary classifier

Since wavelet kernels are slower to compute than other widely used kernels, and the first two points of the above issues rely on a cache rather than the explicit calculation of the kernel function, an efficient implementation should allow a larger than normal cache to achieve a faster execution. However, a larger cache will not decrease the number of iterations or the computational complexity.

3.7 Methodology

3.7.1 Performance Measures

Accuracy is used as a statistical measure of how well a binary classification test correctly identifies or excludes a condition.

That is, the accuracy is the proportion of true results (both true positives and true negatives) in the benchmark collection. It is a parameter of the test (see also Table 3.1).

$$\text{accuracy} = \frac{\text{number of true positives} + \text{number of true negatives}}{\text{numbers of true positives} + \text{false positives} + \text{false negatives} + \text{true negatives}}$$

To verify the impact of the difference in data on the performance variation of the benchmarked methods and further evaluate whether there is significant difference between them, McNemar's significance tests is employed (Everitt, 1992). McNemar's test is a χ^2-based significance test for goodness of fit that compares the distribution of counts expected under the null hypothesis to the observed counts.

n_{00}: number of examples misclassified by both $\hat{f}_A$ and $\hat{f}_B$	n_{01}: number of examples misclassified by $\hat{f}_A$ but not by $\hat{f}_B$
n_{10}: number of examples misclassified by $\hat{f}_B$ but not by $\hat{f}_A$	n_{11}:number of examples misclassified by neither $\hat{f}_A$ nor $\hat{f}_B$

Table 3.2: Contingency table for McNemar's test

n_{00}	$\frac{n_{01}+n_{10}}{2}$
$\frac{n_{01}+n_{10}}{2}$	n_{11}

Table 3.3: Expected counts under the null hypothesis for McNemar's test

Following (Dietterich, 1998) to apply McNemar's test, the benchmark collection is divided into a training set R and a test set T. Both benchmarked algorithms A and B are trained on the training set yielding classifiers $\hat{f}_A$ and $\hat{f}_B$. Then these classifiers are tested on the test set. For each example $x \in T$, it is recorded how it was classified and a contingency table is constructed (Table 3.2, where $n = n_{00} + n_{01} + n_{10} + n_{11}$ is the total number of examples in the test set T).

Under the null hypothesis, the two algorithms should have the same error rate, which means that $n_{01} = n_{10}$. McNemar's test is based on a χ^2 test for goodness-of-fit that compares the distribution of counts expected under the null hypothesis to the observed counts. The expected counts under the null hypothesis are shown in Table 3.3.

The following statistic is distributed (approximately) as χ^2 with 1 degree of freedom; it incorporates a "continuity correction" term (of -1 in the numerator) to account for the fact that the statistic is discrete while the χ^2 distribution is continuous:

$$\frac{(|n_{01} - n_{10}| - 1)^2}{n_{01} + n_{10}}.$$

The null hypothesis may be rejected in favor of the hypothesis that the two

algorithms have different performance when trained on the particular training set R.

3.7.2 Benchmark Collections

For general benchmarking, a data collection has to fulfill the following two conditions:

1. Features must be homogenic: they should represent the same kind of measurements.

2. Subsets of features should be correlated.

While the latter condition is true for most collections, homogenic features are scarce. One example is DNA microarray data. A DNA microarray is a technology used in molecular biology and in medicine. It consists of an arrayed series of thousands of microscopic spots of DNA oligonucleotides, these are the features, each containing picomoles of a specific DNA sequence.

In the experiments the Leukemia DNA microarray data is used[1]. This collection consists of 38 training and 34 testing instances, with 7129 features and a dense feature set.

Madelon is an artificial data set containing data points grouped in 32 clusters placed on the vertexes of a five dimensional hypercube and randomly labeled +1 or -1. The five dimensions constitute 5 informative features. 15 linear combinations of those features were added to form a set of 20 (redundant) informative features. Based on those 20 features one must separate the examples into the 2 classes (corresponding to the ± 1 labels). A number of distractor features called 'probes' was added having no predictive power. The order of the features and patterns were randomized. The collection consists of 2000 training and 1800 test instances.

[1] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#leukemia

	Leukemia	Madelon	Gisette
Number of Features	7129	20	5000
Training Instances	38	2000	6000
Test Instances	34	1800	1000

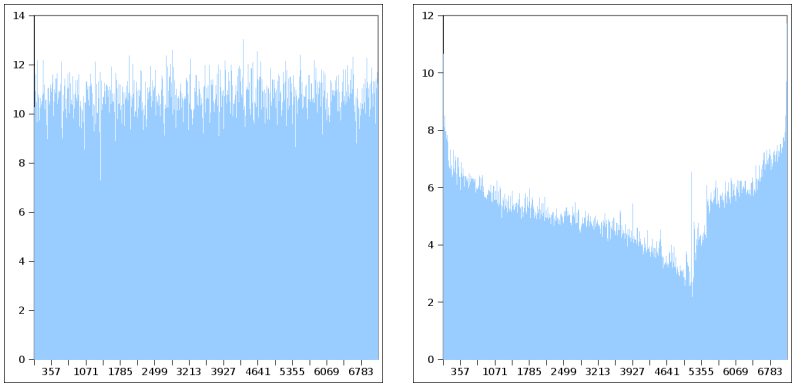
Table 3.4: Results with baseline kernels

The task of Gisette is to discriminate between to confusable handwritten digits: the four and the nine. This is a two-class classification problem with sparse continuous input variables. The digits have been size-normalized and centered in a fixed-size image of dimension 28x28. The feature set consists of the original variables (normalized pixels) plus a randomly selected subset of products of pairs of variables. The pairs were sampled such that each pair member is normally distributed in a region of the image slightly biased upwards. The training set contained 6000 examples and test set 1000, with a total of 5000 features.

Table 3.4 summarizes the test collections. These collections were chosen because they are fairly widely used in benchmarking algorithms that need continuous attributes, hence we believe they demonstrate our ideas well.

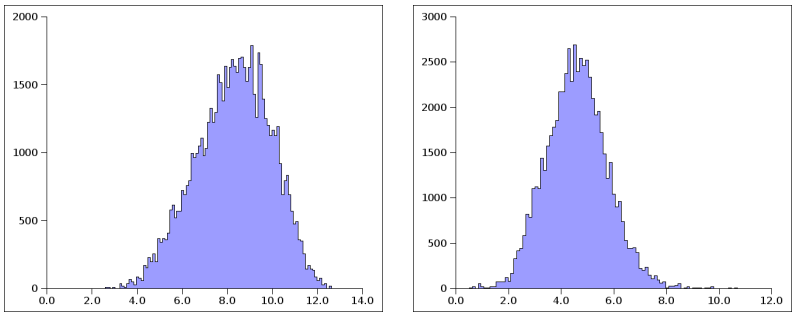
3.8 Experimental Results

Before studying the actual performance of the proposed kernel, we compare the ordination generated by OPTICS with our proposed algorithm (Section 3.8.1). Then we look at classificaton performance (Section 3.8.2), followed by an analysis on parameter sensitivity (Section 3.8.3).



(a) Consecutive distances with OPTICS

(b) Consecutive distances with OHP



(c) Histogram of distances with OPTICS

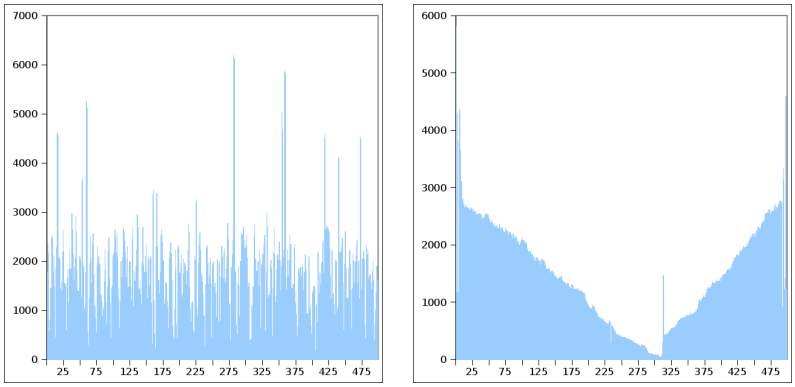
(d) Histogram of distances with OHP

Figure 3.13: The Quality of ordination on the Leukemia data set

3.8.1 Comparison of OPTICS and the Ordination Algorithm

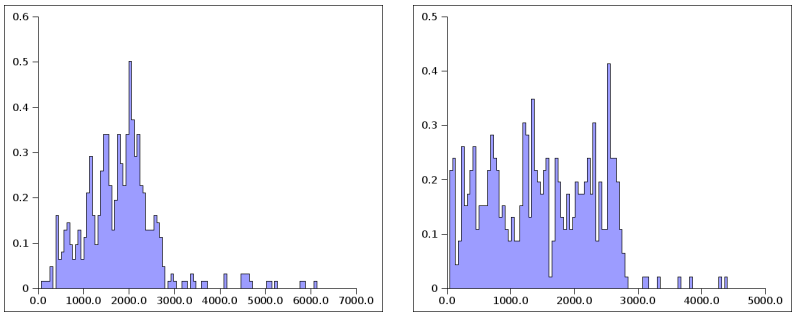
Since OPTICS is parametric, we are interested in whether we are able to replicate or even improve the results with our proposed non-parametric method. We wanted to see how well the algorithms choose the features to be put next to each other. We calculated the consecutive distances, histograms and average distances by the Euclidean distance function. We used only the training part of the data sets to calculate the distances between the features. Table 3.5 summarizes the results.

Figures 3.13(a), 3.13(b), 3.14(a), 3.14(b), 3.15(a), and 3.15(b) plot the consecutive distances between features. The x axis refers to the consecutive features,



(a) Consecutive distances with OPTICS

(b) Consecutive distances with OHP



(c) Histogram of distances with OPTICS

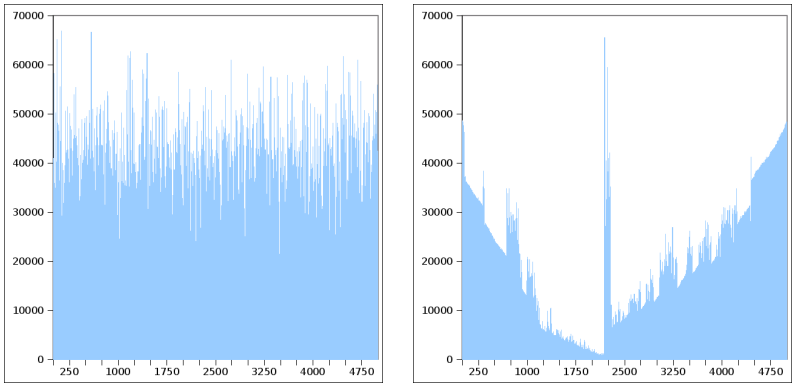
(d) Histogram of distances with OHP

Figure 3.14: The Quality of ordination on the Madelon data set

while the y axis represents the distance between two consecutive features.

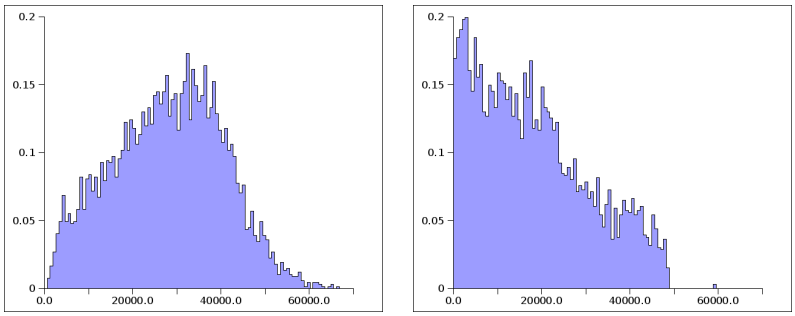
The plot of consecutive distances on the Leukemia data set clearly indicates the suboptimal nature of OHP (Fig. 3.13(b)). In the beginning, the algorithm is able to choose features located nearby, but as the number of choices reduces, the quality of the ordination decreases. However, OPTICS shows an overall bad performance on consecutive distances, as also shown by the histogram and the average distance.

When studying the quality of ordination on the Madelon data set, the result is very similar. OHP initially greedily chooses good candidates, but towards the two ends of the spectrum the quality drops. While the histograms do not show a



(a) Consecutive distances with OPTICS

(b) Consecutive distances with OHP



(c) Histogram of distances with OPTICS

(d) Histogram of distances with OHP

Figure 3.15: The Quality of ordination on the Gisette data set

clear winner, the average distance is lower for OHP by nearly 20 %.

A curious phenomenon is captured in the plot of consecutive distances with OHP ordination on the Gisette data set. There are several local spikes which shows that the greedy algorithm is probably very far off from the optimal solution. However, OHP is still better then OPTICS, the latter method's average distance being one and the half times higher than the former's.

3.8.2 Classification Performance

The benchmarks compare traditional kernels (linear, polynomial, RBF), a wavelet kernel (Zhang, Zhou, and Jiao, 2004), and CSBF kernels with feature ordering

Data Set	OPTICS	OHP
Leukemia	8.275	4.599
Madelon	1817.804	1475.589
Gisette	28030.860	18097.961

Table 3.5: Average distance

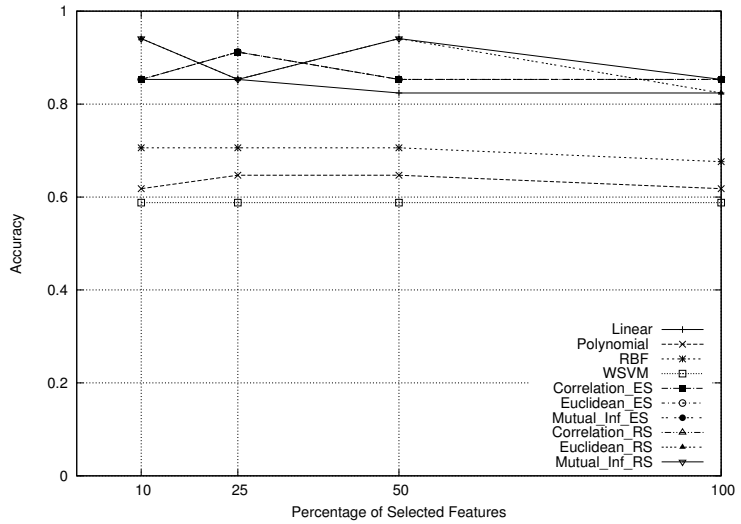


Figure 3.16: Accuracy versus percentage of features, Leukemia data set

performed with three distance functions: correlation, Euclidean, and mutual information. All CSBF kernels were benchmarked with equally spaced and randomly spaced observations.

We used the `libsvm` (Chang and Lin, 2001) library to benchmark the baseline kernels, and implement the suggested kernel. Feature selection was done with χ^2 feature ranking using Weka (Hall et al., 2009), we selected 10 %, 25 %, 50 % and 100 % of the features to understand whether keeping all features would be beneficial to the overall performance. We used only C-SVMs, and for each kernel a wide range of kernel-specific parameters were benchmarked. The parameter C and other

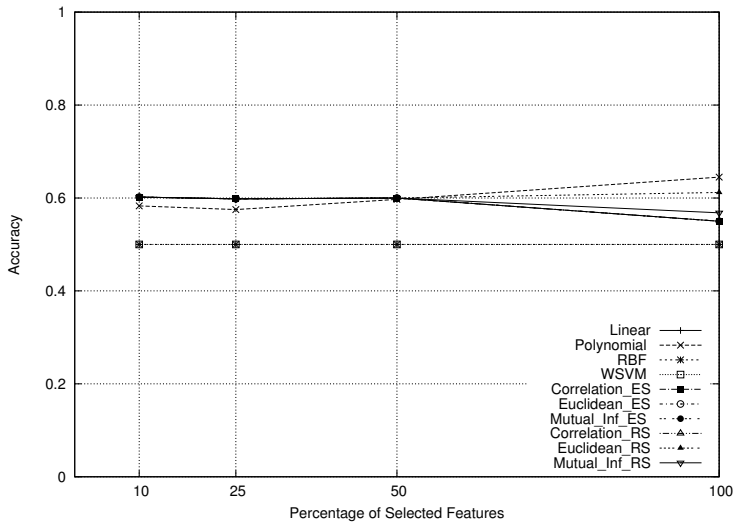


Figure 3.17: Accuracy versus percentage of features, Madelon data set

non-kernel specific parameters were left at default values.

The linear kernel is parameter-free, hence only one run was performed for each dataset. For polynomial kernels, the degree was varied (2 and 3), as well as the offset (0 and 1). RBF kernels converge very slowly in the `libsvm` implementation, therefore only the default parameter value (one over the number of features) and unit γ were benchmarked on all data sets, while a wider range of settings showed the insensitivy of the parameters on smaller collections. WSVM kernels were tested with parameters 1, 2.5, 5 and 10; however, this kernel also proved insensitive to the choice of parameters. All kernels are reported with a result for the best parameter setting.

Figures 3.16, 3.17 and 3.18 summarize the results for the all kernels with the best parameter settings for each kernel. The underlying data can be found in the Appendix.

We benchmarked three different distance functions to reorder the feature set: correlation, mutual information and Euclidean distances. Ordering was performed

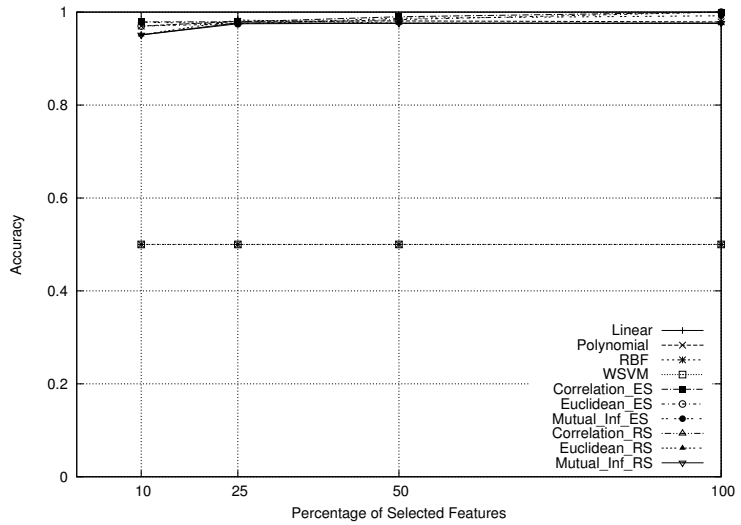


Figure 3.18: Accuracy versus percentage of features, Gisette data set

after feature selection. Benchmarks were conducted with both equally spaced and randomly spaced observations (Es and Rs in the figures, respectively). The following support lengths were benchmarked: 0.5, 1, 2, 5, 10, 20.

Since the previous section showed that the ordination generated by OPTICS is not suitable for the kernel, we only benchmarked CSBF kernels with our proposed ordination algorithm.

As expected, a narrow support will result in identical accuracy as with linear kernels, since a narrow support means that no neighboring features are considered (see the Appendix for detailed data). Madelon is the only exception. The linear kernel converges extremely slowly to a solution on this data set, it needs over a million iterations, and the same holds for wavelet kernels. The difference in accuracy is probably due to numerical instability.

Looking at Figure 3.16, CSBF kernels consistently outperform baseline kernels with both equally spaced and randomly spaced observations. However, the margin compared to a linear kernel is not significant in most cases.

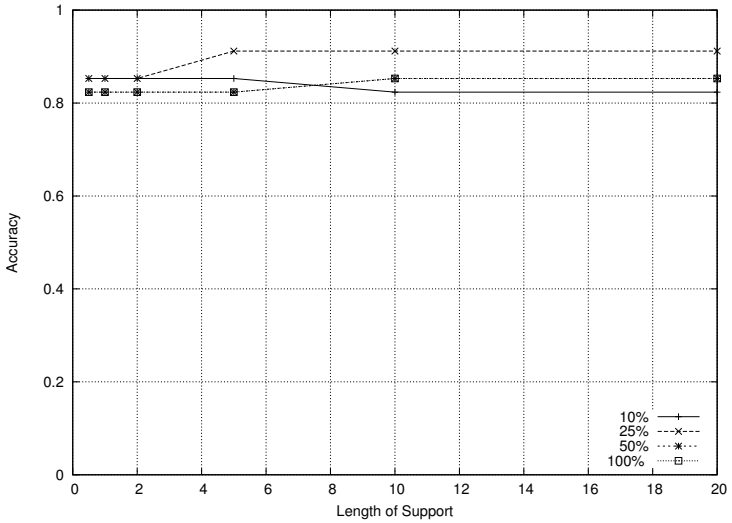


Figure 3.19: Accuracy as the function of the length of support, Leukemia data set

The Madelon data set (Figure 3.17) shows similar results, with the polynomial kernel performing the best by a considerable margin with all features given.

While all results on the Gisette data set are already above 95 % accuracy, CSBF kernels are outstanding with equally spaced observation, reaching full accuracy with all features given. The difference gets significant with at least 50 % of the features selected.

The three collections in the benchmarks appear to be insensitive to feature selection across all tested kernels. For this reason, the advantage of keeping all features and using the dependencies in between them is less apparent.

Kernels with randomly spaced observations tend to peak in performance with a support 10 to 20 wide. It is interesting to note that (Fonseca et al., 2007) have found that Daubechies wavelets with a support of 20 are the most efficient in identifying voice disorders. As opposed to B-spline wavelets, Daubechies wavelets are orthogonal. Nevertheless, the difference between equally spaced observations and randomly spaced observations is either insignificant or considerably worse in

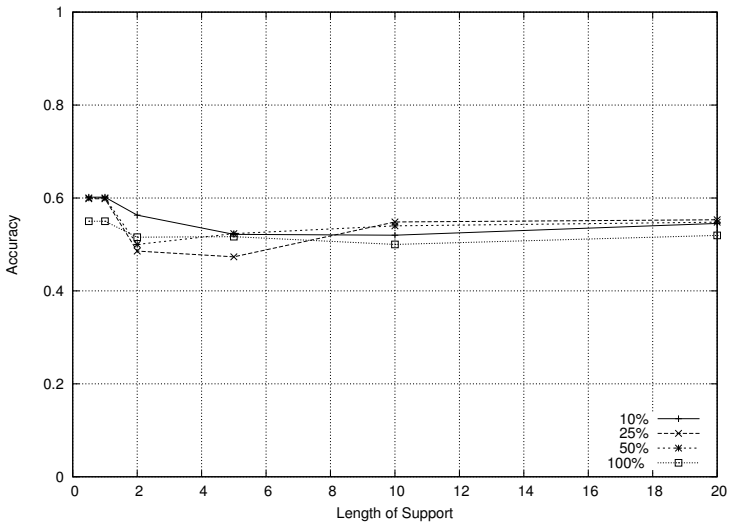


Figure 3.20: Accuracy as the function of the length of support, Madelon data set

the second case.

3.8.3 Parameter Sensitivity

Kernels with randomly spaced observations tend to peak in performance with a support 10 to 20 wide. It is interesting to note that (Fonseca et al., 2007) have found that Daubechies wavelets with a support of 20 are the most efficient in identifying voice disorders. As opposed to B-spline wavelets, Daubechies wavelets are orthogonal.

Figures 3.19, 3.20 and 3.21 plot the accuracy versus the length of support with different percentages of features kept.

As expected, a narrow support will result in identical accuracy as with linear kernels, since a narrow support means that no neighboring features are considered. Madelon is the only exception. The linear kernel converges extremely slowly to a solution on this data set, it needs over a million iterations, and the same holds for wavelet kernels. The difference in accuracy is probably due to numerical instability.

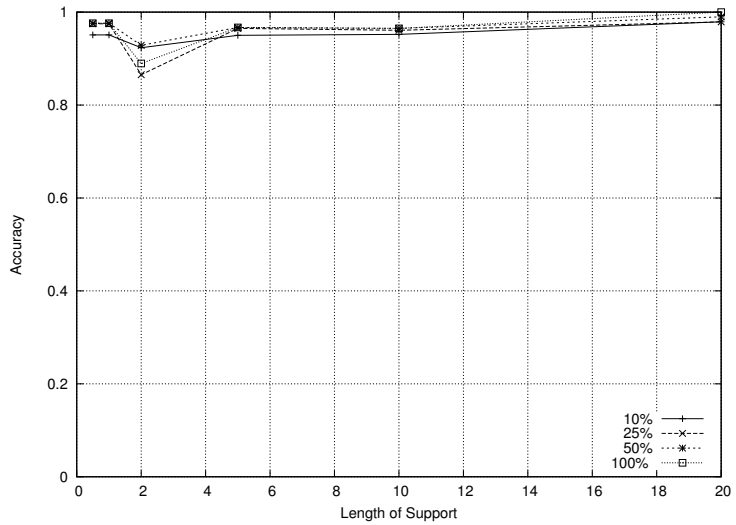


Figure 3.21: Accuracy as the function of the length of support, Gisette data set

The Leukemia data set shows very little change irrespective of how many per cent of the features is being used (Figure 3.19). This is partially due to the small size of the collection.

The performance on Madelon decreases fairly steadily as the length of support increases showing that the noise introduced by the suboptimal nature of the ordination algorithm has a deleterious impact (Figure 3.20).

The Gisette data set, however, shows a slight increase in terms of performance as the length of the support increases (Figure 3.21).

The length of support does not appear to be related to the number of features kept. The general trends are the same, irrespective of the selected features.

Chapter 4

CSBF Kernels for Text Classification

Text classification is a field at the crossroads of machine learning and information retrieval since it shares a number of characteristics of these two fields. Text categorization is a supervised learning task, defined as assigning pre-defined category labels to new documents based on the likelihood suggested by a training set of labeled documents (Yang, 1999). It has raised challenges for statistical learning methods, requiring empirical examination of their effectiveness in solving real-world problems which are often high-dimensional, and have a skewed category distribution.

A text categorization scenario is typically a multi-class classification problem, and these categories are not necessarily stochastically independent. This causes problems with multi-class classification methods as it is a difficult problem to deal with. A text often belongs to more than one category, that is, categories can overlap, hence text classification is a multi-label problem inducing further complexities (Sebastiani, 2002).

Since the early days of the vector space model, it has been debated whether

it is a proper carrier of meaning of texts (Raghavan and Wong, 1986), arguing if distributional similarity is an adequate proxy for lexical semantic relatedness (Budanitsky and Hirst, 2006). With the statistical, i.e. devoid of word semantics approaches there is generally no way to improve both precision and recall at the same time, increasing one is done at the expense of the other (Salton, Wong, and Yang, 1975; van Rijsbergen, 1979). For example, casting a wider net of search terms to improve recall of relevant items will also bring in an even greater proportion of irrelevant items, lowering precision. In the meantime, practical approaches have been proliferating, especially with developments in kernel methods in the last decade (Joachims, 1998; Cristianini, Shawe-Taylor, and Lodhi, 2002; Shawe-Taylor and Cristianini, 2004). Some researchers suggested a more general mathematical framework to accommodate the needs that the vector space model cannot satisfy (van Rijsbergen, 2004; Dominich and Kiezer, 2007). Related concerns have led to the development of a model which transformed the traditional vector space model to a Hilbert space incorporating both referential and distributional patterns of terms while maintaining compatibility with existing algorithms such as kernel methods (Wittek, 2007). The proof-of-concept model performed well in information retrieval, though it had serious issues with scalability (Wittek and Darányi, 2007).

This chapter is organized as follows. Section 4.1 overviews text representation models common in information retrieval and text classification, detailing the steps of pre-processing and some flaws of the vector space model specific to text representation. Feature weighting, feature selection, and feature expansion are discussed in the subsequent sections (Section 4.2 and 4.3), as well linear semantic kernels (Section 4.4), since they can be regarded as a special form of feature expansion. Building on these grounds, Section 4.5 applies CSBF kernels to text classification, paying special attention to feature relatedness and ways to measure

it. Chapter 3 found that data sets that have many relevant, but highly related features benefit the most of the proposed kernels. The experimental results of this chapter (Section 4.6 and 4.7) confirm this finding and show a clear effectiveness gain by using the proposed kernels.

4.1 Text Representation

Generally, there are two key issues involved in text representation, i.e. term type and term weight. Terms can be at different levels, such as syllable, word, phrase, and other sophisticated semantic and/or syntactic representations by exploiting natural language processing knowledge. Section 4.1.1 overviews the mandatory text processing tasks of text classification. The most common text representation model, the vector space model is the point of departure for the proposed semantic kernel, hence it is discussed in detail in Section 4.1.2. Different terms have different importance in a text and thus the term weighting methods assign appropriate weights to them, Section 4.2 mentions some aspects of term weighting.

4.1.1 Prerequisites of Text Representation

Indexing is the assignment of content descriptors (such as class labels, keywords, tags, index terms etc.) to documents in general. In particular, indexing is the process in which natural language texts are being transformed to a mathematical representation of their constituents. Indexing involves the analysis of vocabulary of the text collection, the selection of index terms (terms that are considered as the proxies of meaning), and the assignment of weights to terms in each document.

The selection omits the very common words which do not carry meaning. These frequent and uninformative words are called stop words. Commonly, they are topic-neutral words such as articles, prepositions, articles, pronouns, interjections,

conjunctions, etc (such as, *the, a, of* and so on).

Morphological variants of terms have similar semantic interpretations and can be considered as equivalents for the purpose of text classification. For example, *crow* and *crows* in one document should be represented as a single term (or stem) as they share almost the same semantic interpretation. For this reason, a number of stemming algorithms, or stemmers, have been developed to map several morphological forms of words to a common stem. The industry standard automated stemmer is the Porter stemmer (Porter, 1980). The Porter algorithm uses a suffix list for suffix stripping. Normally it does not matter if the stems are genuine words, thus, *computation*, *computer*, *compute* might be stemmed to *comput*. The result of stemming is not perfect, occasionally words with different meaning are stripped to the same stem. Moreover, the resulting index term vocabulary consists solely of individual words, compound words are disregarded. Lemmatisation is closely related to stemming which also considers the context and the part of speech of the term. However, stemmers are typically easier to implement and run faster, and the reduced accuracy may not matter for some applications, such as information retrieval and text classification.

The necessity of stemming is disputed, sometimes it was reported to decrease effectiveness (Baker and McCallum, 1998). Nevertheless, the common practice is to adopt it, as it reduces the dimensionality of the term space. As stemming maps statistically dependent expressions to one single feature, features will be statistically more independent.

To overcome the flaws of automated stemming, a controlled vocabulary can be used for indexing. Controlled vocabulary schemes mandate the uses of predefined, authorized terms that have been preselected by the designer of the controlled vocabulary as opposed to natural language vocabularies where there is no restriction on the vocabulary that can be used (Manning and Schütze, 1999). A controlled

vocabulary has important advantages such as normalization of indexing concepts, and reduction of noise (Baeza-Yates and Ribeiro-Neto, 1999). However, for general domains a comprehensive vocabulary is difficult to compile. The most complete general domain controlled vocabulary for English is the WordNet database (see Section 4.5.2.1).

It is not always clear what a term (and hence, a feature) should be. The basic units (also called indexing terms) for representing documents can be at different levels, such as sub-word level (syllables), word-level (single token), and multi-word level (phrases, sentences). That is, on the word level, indexing terms refer to single words, while on the multi-word level, indexing terms refer to phrases or sentences (Manning and Schütze, 1999).

The advantage of a sub-word level n-gram representation is its robustness against spelling errors. However, most machine learning algorithms are unable to cope with the rapidly increasing quantity of terms (for example, in 2-gram representation, the word *computer* is represented by using 9 terms as above.), and thus effectiveness of text pre-processing and text classification decreases.

Word-based representation is by far the most common way to represent the content of texts (Baeza-Yates and Ribeiro-Neto, 1999). The advantage of this so-called bag-of-words approach is its simplicity. It ignores the syntactic and semantic structure of the text, only the frequency of a word in a document is recorded (Salton and McGill, 1983).

With the development of recent computational linguistic tools, large quantities of text can be analyzed efficiently with respect to their syntactic structure. Accordingly, some researchers suggested using phrases, rather than individual words, as indexing terms (Lewis, 1992; Schutze, Hull, and Pedersen, 1995). The notion of phrase incorporates syntactic and/or statistical information. From the syntactic aspect, the phrase is constructed according to a grammar of the language (Lewis,

1992). From the statistical aspect, the phrase is composed of a set/sequence of words whose patterns of contiguous occurrence in the collection are statistically significant (Caropreso, Matwin, and Sebastiani, 2001) (see also Section 4.5.2.2). In this thesis, a term may interchangeably refer to a word or a phrase.

There were several attempts to tackle the problem of polysemy and synonymy (Sanderson, 2000; Kehagias et al., 2003). The same word may have a number of senses, and the actual sense is dependent on the context. For example, as a noun, *keep* has 3 senses; and as a verb, it has about 22 senses. Synonymous terms refer to the same or similar senses. Indexing by senses, as opposed indexing by terms, was expected to improve effectiveness. This conjecture was tested in information retrieval (Sanderson, 2000) and text classification (Kehagias et al., 2003), but results fell behind expectations.

Term clustering groups together words with a high degree of pairwise relatedness, so that the groups (or their centroid, or a representative of them) may be used instead of the terms (Lewis, 1992; Baker and McCallum, 1998; Slonim and Tishby, 2001). However, the experimental results showed that term clustering methods did not improve the effectiveness of text classification significantly.

4.1.2 Vector Space Model

The vector space model is a classical and still widespread model of text classification and information retrieval, first introduced in 1975 by Salton et al. (Salton, Wong, and Yang, 1975). The model is based on the assumption that the meaning of texts can be represented by their constituting words, that is, a document is nothing more than a bag of words. The number of dimensions of vector space representation is the same as the total number of different terms in the entire corpus. Each coordinate of the vector stands for a specific term.

The representation of documents ignores any semantic relation between the

index terms, as orthogonal vectors are assigned to the index terms. However, in fact, index terms are not independent.

Given the above representation for any two documents, or for a document and query the inner product of the respective vectors serves as a measure of similarity. The 'closest' documents to a query are retrieved for the user, ranked by their distance from the query. However, the inner product has received several critiques (Wong and Raghavan, 1984), and as modifications have been developed to the vector space model, the similarity measure was no longer an inner product in a mathematical sense (Dominich and Kiezer, 2007). The discrepancy has been rectified by introducing a measure theoretic framework, and the mathematical soundness was proved for various vector space-based models (such as the generalized vector space model (Raghavan and Wong, 1986) and latent semantic indexing (Deerwester et al., 1990), but not necessarily for others. It has been proved that the inner product is not a necessary ingredient of the vector space model, and to avoid any possible flaw in the mathematical framework, in this discussion the similarity measure underlying the vector space model is not considered as an inner product (Dominich and Kiezer, 2007). Instead, the similarity measure, though formally an inner product, will be referred to as a kernel. The inner product satisfies the conditions of Mercer's theorem (see 2.2.6). Regarding the similarity measure as a kernel enables a mathematically sound treatment of similarity even with the extensions of the vector space model.

Furthermore, for a particular document the representation is typically extremely sparse, having only relatively few (approximately 1 to 5 % according to (Berry, Drmac, and Jessup, 1999)) nonzero entries. Thus very fast algorithms can be deployed for machine learning.

4.2 Feature Weighting and Selection in Text Representation

Weights usually range between 0 and 1. As a special case, binary weights may be used (1 denoting presence and 0 absence of the term in the document); whether binary or non-binary weights are used depends on the classifier learning algorithm used (Sebastiani, 2002). In the case of non-binary indexing, for determining the weight x_{kj} of term f_k in document x_j is normally a composition of local and global weighting (Salton and Buckley, 1988).

The local part is the immediate relation between a document and an indexing term. The most common weighting is the frequency of the term in the document or its logarithm ($\log(f_{ij}+1)$). This weighting embodies the intuition that the more often a term occurs in a document, the more it is representative of its content.

A common global weighting is the inverse document frequency $\text{idf}_\text{i} = \frac{N}{N_i}$, that is the total number of documents divided by the number of documents containing the term i. That is, the more documents a term occurs in, the less discriminating it is.

In order for the weights to fall in the $[0, 1]$ interval and for the documents to be represented by vectors of equal length, the weights resulting from tfidf are often normalized by cosine normalization, given by:

$$\bar{x_{kj}} = \frac{x_{kj}}{\sqrt{\sum_{i=1}^{M} x_{ij}^2}}.$$

Although normalized tfidf is the most popular one, other indexing functions have also been proposed. This section only offers a glimpse into term weighting, see (Salton and Buckley, 1988; Leopold and Kindermann, 2002; Lan et al., 2009) for more detailed discussions.

As in other domains, text categorization also benefits from feature selection,

Function	Denoted by	Mathematical form
Information gain	$\mathrm{IG}(f_k, c_i)$	$P(c_i \mid f_k)$
Mutual information	$\mathrm{MI}(f_k, c_i)$	$\sum_{c\in\{c_i,\hat{c}_i\}} \sum_{f\in\{f_k,\hat{f}_k\}} P(f,c) \log \frac{P(f,c)}{P(f)P(c)}$
χ^2	$\chi^2(f_k, c_i)$	$\frac{\lvert f_r\rvert[P(f_k,c_i)P(\hat{f}_k,\hat{c}_i)-P(f_k,\hat{c}_i)P(\hat{f}_k,c_i)]^2}{P(f_k)P(\hat{f}_k)P(c_i)P(\hat{c}_i)}$
NGL coefficient	$NGL(f_k, c_i)$	$\frac{\sqrt{\lvert T_r\rvert}[P(f_k,c_i)P(\hat{f}_k,\hat{c}_i)-P(f_k,\hat{c}_i)P(\hat{f}_k,c_i)]}{\sqrt{P(f_k)P(\hat{f}_k)P(c_i)P(\hat{c}_i)}}$
Relevancy score	$RS(f_k, c_i)$	$\log \frac{P(f_k\mid c_i)+d}{P(\hat{f}_k\mid c_i)+d}$
Odds ratio	$OR(f_k, c_i)$	$\frac{P(f_k\mid c_i)(1-P(f_k\mid \hat{c}_i))}{1-P(f_k\mid c_i))P(f_k\mid \hat{c}_i)}$
GSS coefficient	$GSS(f_k, c_i)$	$P(f_k, c_i)P(\hat{f}_k, \hat{c}_i) - P(f_k, \hat{c}_i)P(\hat{f}_k, c_i)$

Table 4.1: Most important functions used for space reduction purposes in text representation

though it has been shown that this does not result in significant gain in efficiency (Yang and Pedersen, 1997). Since document vector spaces are extremely sparse, the compactness of a reduced feature set is less important, while retaining a higher number of features does have a positive impact on efficiency (Joachims, 1998). Table 4.1 summarizes the most important functions used in feature selection for text categorization (Sebastiani, 2002). In the table, f_k denotes a feature and c_i denotes a class. A hat over a feature indicates the abscence of the feature, and a hat over a class means all the other classes.

4.3 Feature Expansion in Text Representation

In order to eliminate the bottleneck of the traditional BOW representation, previous approaches in term expansion enriched this convention by external lexical resources such as WordNet.

As a first step, these methods generate new features for each document in the data set. These new features can be synonyms or homonyms of document terms as in (Hotho, Staab, and Stumme, 2003; Rodriguez and Hidalgo, 1997), or expanded

features for terms, sentences and documents as in (Gabrilovich and Markovitch, 2005).

Then, the generated new features replace the old ones or are appended to the document representation, and construct a new vector representation $\hat{\mathbf{x}}_i$ for each text document. The similarity measure of document pairs is defined as:

$$S(\hat{\mathbf{x}}_i, \hat{\mathbf{x}_j}) = \frac{\hat{\mathbf{x}}_i \hat{\mathbf{x}}_j}{|\hat{\mathbf{x}}_i||\hat{\mathbf{x}}_j|}. \tag{4.1}$$

Several distributional criteria have been used to select terms related to the query. For instance, (Robertson, 1990) proposed the principle that the selected terms should have a higher probability in the relevant documents than in the irrelevant documents. Others examined the impact of determining expansion terms using a minimal spanning tree and some simple linguistic analysis (Smeaton and van Rijsbergen, 1983).

To date, the work on integrating semantic background knowledge into information retrieval and related tasks is intensive, but the results are not necessarily convincing (Voorhees, 1994). Some researchers have successfully integrated WordNet as a resource for a document categorization task (Rodriguez and Hidalgo, 1997; Ureña López, Buenaga, and Gómez, 2001). Others found that WordNet synsets as features for document representation may eventually decrease performance if word sense disambiguation is not performed (Gonzalo et al., 1998; Dave, Lawrence, and Pennock, 2003)

By integrating WordNet-based knowledge into text clustering, plus investigating word sense disambiguation strategies and feature weighting schema by considering the hypernym relations from WordNet, experimental results on the Reuters corpus showed improvements compared with the best baseline (Hotho, Staab, and Stumme, 2003). Also, the enrichment strategy to append or replace document terms with their hypernyms or synonyms is overly simplistic.

Despite the large number of studies, a crucial issue that has not been directly

addressed is whether all the expansion terms selected in one way or another are truly useful for the retrieval. One was usually concerned with the global impact of a set of expansion terms. This might suggest that most expansion terms are useful (Peat and Willett, 1991). The question whether the expansion terms are equally important has been addressed in certain domains (Abdou and Savoy, 2008) and using heuristics ; a generic approach is lacking.

4.4 Linear Semantic Kernels

Kernel methods offer a completely different strategy to feature enrichment (Section 2.2.6). For textual data, linear kernels have been investigated. Any linear kernel for texts is characterized by

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i' S' S \mathbf{x}_j,$$

where S is an appropriately shaped matrix (see Section 3.3 on kernel properties). The presence of S changes the orthogonality of the vector space model, as this mapping should introduce term dependence. A recent attempt tries to manually construct S with the help of a lexical resource (Siolas and d'Alché Buc, 2000). The entries in the symmetric matrix S express the semantic proximity between the terms i and j. Such a semantic kernel is designed as a combination of the RBF kernel with the term proximity matrix S. Entries in this matrix are inversely proportional to the length of the WordNet hierarchy path linking the two terms. The performance, measured over the 20NewsGroups corpus, shows an improvement of 2 % over the the basic vector space method. Moreover, the semantic matrix S is almost fully dense, hence computational issues arise.

An early attempt to overcome the untenable orthogonality assumption of the vector space model was proposed under the name of generalized vector space model (Wong, Ziarko, and Wong, 1985). The article which proposed the model did

not provide empirical results (Wong, Ziarko, and Wong, 1985), and since then the model has been regarded of great theoretical importance with less impact on actual applications.

The model takes a distributional approach, focusing on term co-occurrences. The underlying assumption is that term correlations are captured by the co-occurrence information. That is, two terms are semantically related if they co-occur often in the same documents (Wong, Ziarko, and Wong, 1985). By eliminating orthogonality, documents can be seen as similar even if they do not share any terms.

The term co-occurrence matrix is XX', hence the model takes X' as the semantic proximity matrix S. The computational needs, however, are tremendous, if the dimensions of X are considered. Moreover, the co-occurrence matrix is not sparse anymore.

Latent semantic indexing (or latent semantic analysis) was another attempt to bring more linguistic and psychological aspects to language processing via a kernel (Deerwester et al., 1990). Conceptually, latent semantic indexing is similar to the generalized vector space model, it measures semantic information through co-occurrence analysis in the corpus. From the algorithmic perspective it is an enormous problem that textual data have a large number of relevant features. This results in huge computational needs and the classification models may overfit the data. The number of features can be reduced by multivariate feature extraction methods. In latent semantic indexing, the dimension of the vector space is reduced by singular value decomposition (Section 2.1.2.2).

Using rank reduction to get the so-called feature space, terms that occur very often together in the same documents are merged into a single dimension of the feature space. The diversity of expressions and the minor differences in senses are lost. In return, theoretically, those few hundred concepts, which make up the meaning of the documents, can be identified easily. The dimensions of the reduced

space correspond to the axes of greatest variance.

Using the notation of Section 2.1.2.2, the matrix U, and in particular $U'_k = I_k U'$, where I_k is the identity matrix with only the first k diagonal elements nonzero, are projection operators to the feature space, the projection of a document vector is $I_k U' \mathbf{x}_j = U'_k \mathbf{x}_j$. The document vectors are projected into the subspace spanned by the first k singular vectors of the feature space. By choosing $S = I_k U'$ the resulting kernel is indeed a linear kernel (Cristianini, Shawe-Taylor, and Lodhi, 2002).

$$K(\mathbf{x}_i, \mathbf{x}_j) = (I_k U' \mathbf{x}_i)' I_k U' \mathbf{x}_j = \mathbf{x}'_i U I_k U' \mathbf{x}_j = \mathbf{x}'_i S' S \mathbf{x}_j.$$

The projection of the generalized vector space model can be written in a similar fashion (Shawe-Taylor and Cristianini, 2004).

$$K_{GVSM}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}'_i X X' \mathbf{x}_j = \mathbf{x}'_i U \Sigma^2 U' \mathbf{x}_j.$$

Apparently, the generalized vector space model does not apply dimensionality reduction, and does not use orthonormal projections of the data (the presence of the matrix Σ ensures the projections are not orthonormal).

For latent semantic indexing by dual representation the kernel matrix is

$$K = (U'_k X)' U'_k X = (I_k U' X)' I_k U' X = X' U I_k U' X = V \Sigma U' U I_k U' U \Sigma V' = V \Sigma_k^2 V'.$$

The new kernel matrix can be obtained directly from K by applying an eigenvalue decomposition of K (Cristianini, Shawe-Taylor, and Lodhi, 2002).

The computational complexity of performing an eigenvalue decomposition on the kernel matrix is a major drawback of latent semantic indexing. Although the size of the matrix is a small fraction of the initial space, it is usually no longer sparse, hence information retrieval or any other operation is much slower. Unfortunately, none of the above semantic linear kernels improved significantly the effectiveness of text classification (Cristianini, Shawe-Taylor, and Lodhi, 2002; Basili, Cammisa, and Moschitti, 2005).

4.5 A Different Approach to Text Representation

As Basili et al. pointed out, when a similarity function is extended to use external prior knowledge (that is, a lexical resource), the critical issues are the methods and conditions to integrate such knowledge (Basili, Cammisa, and Moschitti, 2005). This section proposes a novel method for this integration and analyzes its assumptions.

Section 4.5.1 details the background of mapping documents to the L_2 space and also proves that this representation fits into the framework of kernel methods and support vector machines. The inner product of the L_2 space offers a way to improve the effectiveness of text classification by incorporating the semantic relatedness of terms using the algorithm proposed in Section 3.5.

4.5.1 Semantic Kernels in the L_2 Space

While linear semantic kernels failed to improve overall effectiveness of text classification (see Section 2.2.6), non-linear kernels have not been investigated yet. This section introduces a new, non-linear semantic kernel. The proposed model of document representation is based on the classical vector space model. It is non-classical inasmuch as the coordinates of real-valued document vectors are interpreted as a subset of the range of continuous functions. The continuity of semantic content is a prerequisite for this model, provided by semantic term ordering (see Section 3.5).

In order to reproduce the continuity of semantic content, one has to deal with the following problem. A word is a hypernym if its meaning encompasses the meaning of another word of which it is a hypernym; it is more generic or broader than another given word. Sometimes hypernymy is referred to as the "is-a" relation . Its opposite relation is hyponymy. A word is a hyponym (Lyons, 1977) if its semantic range is included within that of another word; it is also referred to

as the "instance-of" relation. For example, *ship* is an instance of *vessel*, so *ship* is a hyponym thereof, whereas *icebreaker* is a *ship*, *ship* here being a hypernym to *icebreaker*. For the most comprehensive general domain ontology with a hypernym hierarchy for English, see the WordNet database (Fellbaum, 1998).

The assignment of canonical basis vectors to index terms is arbitrary in case of the classical vector space model. As the proposed kernel heavily utilizes term interdependence, such arbitrary assignment should be avoided. Instead, the ordering of index terms arranges them (see Section 3.5) , with occasional gaps between "islands of similar meaning". Related words are assigned to subsequent vectors of the canonical base. Secondly, consider the hypernymic terms *vessel*, *ship*, and *icebreaker*.

4.5.2 Measuring Semantic Relatedness

Since the distance between index terms is crucial for the proposed kernel, we review the core ideas in quantitizing semantic similarity. Several methods have been proposed for measuring similarity. One of such early proposals was the semantic differential (Osgood, 1952; Osgood, Suci, and Tannenbaum, 1957) which analyzes the affective meaning of terms into a range of different dimensions with the opposed adjectives at both ends, and locates the terms within semantic space (Kozima and Furugori, 1993).

Semantic similarity as proposed by Miller and Charles is a continuous variable that describes the degree of synonymy between two terms (Miller and Charles, 1991). They argue that native speakers can order pairs of terms by semantic similarity, for example *ship-vessel, ship-watercraft, ship-riverboat, ship-sail, ship-house, ship-dog, ship-sun*. This concept may be extended to quantify relations between non-synonymous but closely related terms, for example *airplane-wing* (Jarmasz and Szpakowicz, 2003). Semantic similarity is closely related to the concept of

semantic relatedness. There is some overlap in their meanings and they are used interchangeably in certain contexts. Semantically dissimilar entities may be semantically related, for example, *tree* and *shade* (Mohammad and Hirst, 2005). In this case the two entities are not similar, but are related by some relationship. Two entities are semantically related if they are semantically similar or share any other classical or non-classical relationships (Kehagias et al., 2003). In fact, semantic similarity is a subset of semantic relatedness.

Semantic relatedness is defined between senses of terms. Given a relatedness formula $\text{rel}(s_1, s_2)$ between two senses s_1 and s_2, term relatedness between two terms f_1 and f_2 can be calculated as

$$\text{rel}(f_1, f_2) = \max_{s_1 \in \text{sen}(f_1), s_2 \in \text{sen}(f_2)} \text{rel}(s_1, s_2), \tag{4.2}$$

where $\text{sen}(t)$ is a set of senses of term t (Budanitsky and Hirst, 2006).

The concept of semantic distance has been used in the context of both semantic relatedness and semantic similarity. In the former aspect, it is the inverse of semantic relatedness, while in the latter, it is the inverse of semantic similarity (Mohammad and Hirst, 2005).

Automated systems assign a score of semantic relatedness to a given pair of terms calculated from a relatedness measure. The absolute score itself is typically irrelevant on its own, what is important is that the measure assigns a higher score to term pairs which humans think are more related and comparatively lower scores to term pairs that are less related (Mohammad and Hirst, 2005).

Terms in a language are organized by two kinds of relationship. Syntagmatic similarity is based on co-occurrence data extracted from corpora (Church and Hanks, 1990). Semantic similarity is based on association data extracted from thesauri (Morris and Hirst, 1991) psychological experiments (Osgood, 1952), and so on (Kozima and Furugori, 1993).

The best known theories of word semantics fall in three major groups:

1. "Meaning is use" (Wittgenstein, 1967): habitual usage provides indirect contextual interpretation of any term; frequency of use expresses aspects of a conceptual hierarchy. In terms of logical semantics, one regards document groups as value extensions (classes) and index terms as value intensions (properties) of a (semantic) function 'f' (Carnap, 1947). Extensions and intensions are inversely proportional: the more properties defined, the less entities they apply to - there are more flowers in general than tulips in particular, for instance (Lyons, 1977). The same applies to Boolean information retrieval using the AND logical operator: more frequent, generic index terms with less properties retrieve larger sets than specific terms with more developed descriptions, hence term occurrence relates to conceptual hierarchy.

2. "Meaning is change": the stimulus-response theory (Bloomfield, 1933) and the biological theory of meaning (von Uexküll, 1982) both stress that the meaning of any action is its consequences.

3. "Meaning is equivalence": referential (Frege, 1948; Peirce, 1955) or ostensional theories of meaning suggest that 'X = Y for/as long as Z' (Harnad, 1987).

Point 2 refers to theories which assign a temporal structure to word meaning, they are not discussed here. Measures that rely on distributional measures and those that use knowledge-rich resources both exist, and they have been individually shown to good quantifiers of term similarity each (Mohammad and Hirst, 2005). This section reviews lexical resources (Section 4.5.2.1) and measures based on them (Section 4.5.2.2), while a brief introduction to distributional measures is provided in the Future Work section.

4.5.2.1 Lexical Resources

In computer science, an ontology defines the concepts, relationships, and other distinctions that are relevant for modeling a domain (Gruber, 1995). A slightly different, but related concept is a lexical resource: the purpose is not modeling, but capturing semantic relations among terms. Such a resource necessarily entails some sort of world view with respect to a given domain. This is often conceived as a set of concepts, their definitions and their inter-relationships; this is referred to as a conceptualization (Uschold and Gruninger, 1996).

The following types of resources are commonly used in measuring semantic similarity between terms.

1. Dictionary (Lesk, 1986; Kozima and Furugori, 1993);
2. Semantic networks, such as WordNet (Fellbaum, 1998);
3. Thesauri modeled on Roget's Thesaurus (Morris and Hirst, 1991; Jarmasz and Szpakowicz, 2003).

This section briefly describes the above types of sources of lexical knowledge, and offers some criteria to evaluate them.

A dictionary is essentially a closed paraphrasing system of natural language. Each of its entries is defined by a phrase which is composed of the entries and their derived forms: a dictionary is like a network of terms (Kozima and Furugori, 1993).

WordNet is a unique combination of a controlled vocabulary, a dictionary, and a thesaurus. The noun and verb portion of WordNet has a fairly rich connectivity as well as comprehensiveness (Sussna, 1993). The WordNet term nodes are connected by semantic relations. The relations are as follows (Fellbaum, 1998).

- synonymy (has same meaning as; intranode);
- hypernymy (is a);

- hyponymy (has instance);
- holonymy (is part of, is substance in, is member of);
- meronymy (has part, contains substance, has member);
- antonymy (is complement of; self-inverse).

Hypernymy and hyponymy are the strictly hierarchical links (Figure 4.1). The holonymy/meronymy relations can also be considered vertical relations. Vertical relations are asymmetrical and ordered items. Synonymy and antonymy are horizontal, symmetrical, non-ordering relations (and naturally are non-hierarchical) (Fellbaum, 1998).

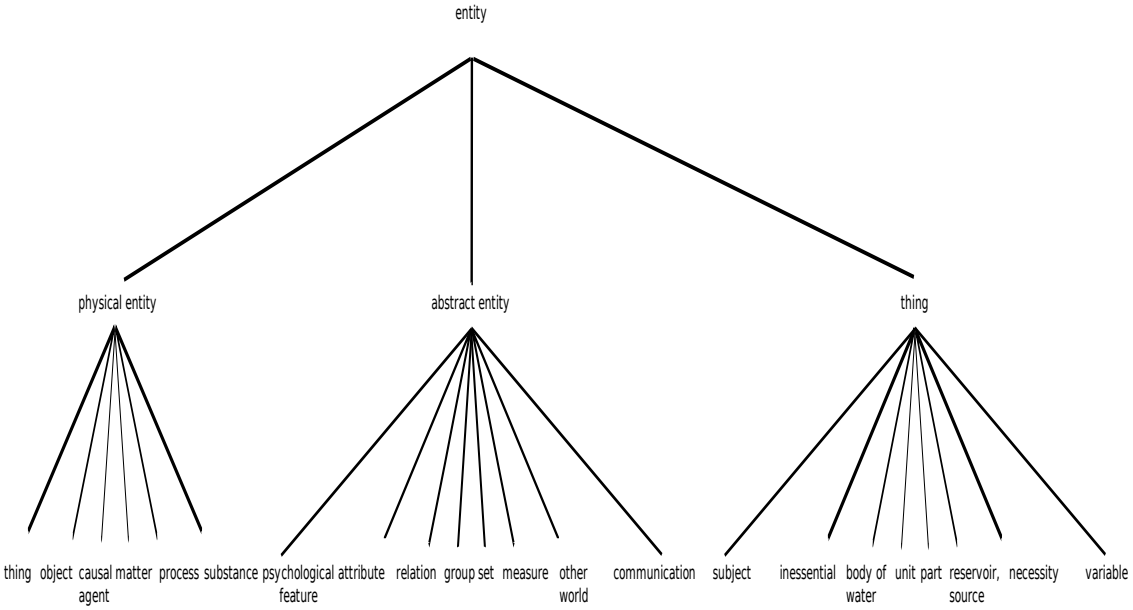


Figure 4.1: First three levels of the WordNet hypornymy hierarchy.

Roget's Thesaurus is based on a well-constructed concept classification, and its entries were written by professional lexicographers (Berrey and Carruth, 1962). Roget's does not have some of the shortcomings of WordNet, such as the lack

of links between parts of speech and the absence of topical groupings (Jarmasz and Szpakowicz, 2003). The Thesaurus can link the noun *bank*, the business that provides financial services, and the verb *invest*, to give money to a bank to get a profit, by placing them in a common head 784 Lending. This type of connection cannot be described using the semantic relations of WordNet. While an English speaker can identify a relation not provided by WordNet, in certain cases the same does not apply to computer systems (Jarmasz and Szpakowicz, 2003).

Eight classes head the taxonomy. The first three, Abstract Relations, Space and Matter, cover the external world. The remaining ones, Formation of ideas, Communication of ideas, Individual volition, Social volition, Emotion, Religion and Morality deal with the internal world of human beings. A path in Roget's Thesaurus always begins with one of the classes. It branches to one of the 39 sections, then to one of the 79 sub-sections, then to one of the 596 head groups and finally to one of the 990 heads (Berrey and Carruth, 1962). Where applicable, categories are organized into antonym pairs. For example, category 407 is *Life*, and category 408 is *Death*. Each head is divided into paragraphs grouped by parts of speech: nouns, adjectives, verbs and adverbs. Finally a paragraph is divided into semicolon groups of semantically closely related terms (Jarmasz and Szpakowicz, 2003). In addition, a semicolon group may have cross-references or pointers to other related categories or paragraphs.

The thesaurus has an index, which allows for retrieval of terms related to a given term. For each entry, a list of terms suggesting its various distinct subsenses is given, plus a category or paragraph number for each of these as well. An index entry may be a pointer to a category or paragraph if there are no subsenses to be distinguished. In this thesaurus, physical closeness has some importance, as can be clearly seen from the hierarchy, but terms in the index of the thesaurus often have widely distributed categories, and each category often points to a widely distributed

selection of categories (Morris and Hirst, 1991).

While there seems to be a consensus on which lexical resources should be used in the general domain (Budanitsky and Hirst, 2006), domain-specific resources may vary in quality. Hence the demand for a set of objective criteria for evaluating such resources emerges. Gruber suggested some criteria for evaluating ontologies (Gruber, 1995):

1. Clarity: An ontology should effectively communicate the intended meaning of defined terms. Definitions should be objective and formalism is a means to this end.

2. Coherence: coherence refers to the incapability of getting contradictory conclusions simultaneously from valid input data. An ontology is semantically consistent if and only if its definitions are semantically consistent. (Gomez-Perez, 1995).

3. Extendibility: An ontology should be designed to anticipate the uses of the shared vocabulary. It should offer a conceptual foundation for a range of anticipated tasks and the representation should be crafted so that one can extend and specialize the ontology monotonically.

4. Minimal encoding bias: The conceptualization should be specified at the knowledge level without depending on a particular symbol-level encoding. An encoding bias results when a representation choices are made purely for the convenience of notation or implementation.

In addition, Gomez-Perez emphasizes the importance of completeness and conciseness (Gomez-Perez, 1995). Completeness refers to the extension, degree, amount or coverage to which the information in a user-independent ontology covers the information of the real world. A definition is complete if all that is supposed to be in the

definition is in the definition or can be inferred from the axioms. Conciseness refers to if all the information gathered in the ontology is useful and precise. Conciseness does not necessarily imply absence of redundancies (Gomez-Perez, 1995).

4.5.2.2 Lexical Resource-Based Measures

Several researchers have done an extensive survey of the various lexical resource-based measures, their comparisons with human judgment on selected term pairs, and their efficacy in applications such as spelling correction and term sense disambiguation (Budanitsky and Hirst, 2001; Budanitsky and Hirst, 2006). This section provides only a brief summary of the most important measures of similarity.

All approaches to measuring semantic relatedness that use a lexical resource regard the resource as a network or a directed graph, making use of the structural information embedded in the graph (Jiang and Conrath, 1997; Budanitsky and Hirst, 2006).

One of the earliest measures is the edge counting method. The shortest path in the network between the two target terms is determined. The more edges there are between two terms, the more distant they are. If all the edges are of equal length, then the number of intervening edges is a measure of the distance (Rada et al., 1989). In determining the overall edge based similarity, most methods just simply sum up all the edge weights along the shortest path. For a hierarchical network, the distance should satisfy the properties of a metric in a mathematical sense (Rada et al., 1989):

1. $d(f_i, f_i) = 0$,
2. $d(f_i, f_j) \geq 0$,
3. $d(f_i, f_j) = (f_j, f_i)$,
4. $d(f_i, f_k) \leq d(f_i, f_j) + d(f_j, f_k)$,

for any given terms f_i, f_j, f_k.

Morris and Hirst used Roget's Thesaurus as knowledge base for determining whether or not two terms are semantically related (Morris and Hirst, 1991). Two terms were considered related if their base forms satisfied any one of the following conditions:

1. They have a category in common in their index entries;
2. One has a category in its index entry that contains a pointer to a category of the other;
3. One is either a label in the other's index entry or is in a category of the other;
4. They are both contained in the same subcategory;
5. They both have categories in their index entries that point to a common category.

This method can capture almost all types of semantic relations, such as paraphrasing by superordinate (for example *cat/pet*), systematic relation (for example *north/east*, and non-systematic relation (for example *tree/shade*) (Morris and Hirst, 1991).

Jarmasz and Szpakowicz also implemented a measure based on Roget's Thesaurus (Jarmasz and Szpakowicz, 2003). Given two terms, they looked up in the index their references that point to the Thesaurus. Then they calculated all paths between references using Roget's taxonomy. The distance equals the number of edges in the shortest path. Path lengths are as follows.

- Length 0: the same semicolon group. *journey's end - terminus*
- Length 2: the same paragraph. *devotion - abnormal affection*
- Length 4: the same part of speech. *popular misconception - glaring error*

- Length 6: the same head. *individual - lonely*
- Length 8: the same head group. *finance - apply for a loan*
- Length 10: the same sub-section. *life expectancy - herbalize*
- Length 12: the same section. *Creirwy (love) - inspired*
- Length 14: the same class. *translucid - blind eye*
- Length 16: in the Thesaurus. *nag - like greased lightning*

To convert the distance measure to a similarity measure, one may simply subtract the path length from the maximum possible path length ((Resnik, 1995):

$$\mathrm{sim}(f_1, f_2) = 2d_{max} - \min_{s_1 \in \mathrm{sen}(f_1) s_2 \in \mathrm{sen}(f_2)} \mathrm{len}(s_1, s_2)$$

where d_{max} is the maximum depth of the network, and the len function is the simple calculation of the shortest path length.

The simple edge-counting relies on an ideal hierarchy with edges of equal length. In hierarchical networks based on natural languages, the edges are not of the same length. It is therefore necessary to consider that the edge connecting the two nodes should be weighted. In general, the edges in this type of hierarchy tend to grow shorter with depth (McHale, 1998). The modifications include the density of the subhierarchies, the depth in the hierarchy where the term is found, the type of links, and the information content of the nodes subsuming the term (Jiang and Conrath, 1997).

The use of density is based on the observation that terms in a more dense part of the hierarchy are more closely related than terms in sparser areas (Richardson and Smeaton, 1995). The observation about density may be an overgeneralization. In Roget's Thesaurus, category 277 *ship/boat* has many more terms than category

372 *blueness*. That does not mean that *kayak* is more closely related to *tugboat* than *sky blue* is to *turquoise* (McHale, 1998).

Depth in the hierarchy is another attribute often used. It can be argued that the distance shrinks as one descends the hierarchy, since differentiation is based on finer details (Jiang and Conrath, 1997). It may be more useful in the deep hierarchy of WordNet than it is in Roget's where the hierarchy is fairly flat and uniform (McHale, 1998).

Type of link can be viewed as the relation type between nodes. In many networks the hyponym/hypernym link is the most common concern. Many edge-based models consider only the IS-A link hierarchy (Rada et al., 1989; Lee, Kim, and Lee, 1993). In fact, other link types/relations, such as meronym/holonym, should also be considered as they would have different effects in calculating the edge weight (Banerjee and Pedersen, 2002). If the target path consists of edges that belong to a number of such relations, the target terms are likely to be more distant (Hirst and St-Onge, 1998).

Sussna considered the first three factors in a weight determination scheme. The weight between two nodes s_1 and s_2 is calculated as follows (Sussna, 1993).

$$\mathrm{wt}(s_1, s_2) = \frac{\mathrm{wt}(s_1 \rightarrow_r s_2) + \mathrm{wt}(s_2 \rightarrow_{r'} s_1)}{2 \max\{\mathrm{depth}(s_1), \mathrm{depth}(s_2)\}}$$

given

$$\mathrm{wt}(c \rightarrow_r) = \max_r - \frac{\max_r - \min_r}{\mathrm{edges}_r(c)}$$

where $\rightarrow_r$ is a relation of type r, $\rightarrow_{r'}$ is its reverse, $\max_r$ and $\min_r$ are the maximum and minimum weights possible for a specific relation type r respectively, and $\mathrm{edges}_r(x)$ is the number of relations of type r leaving node x (Sussna, 1993). The synonym relation was assigned a zero weight, while the nine internode relation types had weight ranges as follows: hypernymy, hyponymy, holonymy, and meronymy all have weights ranging from 1 to 2, and antonymy arcs were assigned the value 2.5, respectively (Sussna, 1993).

Applying this distance formula to a term sense disambiguation task, (Sussna, 1993) showed an improvement where multiple sense terms have been disambiguated by finding the combination of senses from a set of terms which minimizes total pairwise distance between senses. The performance was found to be robust under a number of perturbations; however, depth factor scaling and restricting the type of link to a strictly hierarchical relation impaired performance (Sussna, 1993).

Such measures are highly dependent on the subjectively pre-defined network hierarchy. Since the original purpose of the design of the WordNet or Roget's Thesaurus was not for similarity computation purpose, some local network layer constructs may not be suitable for direct distance manipulation (Jiang and Conrath, 1997).

The original Lesk algorithm performed word sense disambiguation by comparing the definition of each sense of a word in a phrase to the glosses of every sense of each word in a phrase, where the definitions were extracted from the Oxford Advanced Learner's Dictionary of Current English (Lesk, 1986) . A word was assigned the sense whose definition shared the largest number of words in common with the glosses of the other words. For example, in *time flies like an arrow*, the algorithm compared the glosses of *time* to all the glosses of *fly* and *arrow*. Next it compared the definition of *fly* with those of *time* and *arrow*, and so on. Banarjee and Pedersen modified Lesk's approach to take advantage of the inter–connected set of relations among synonyms that WordNet offers (Banerjee and Pedersen, 2002). While Lesk's algorithm restricted the comparisons to the glosses of the words being disambiguated, this latter approach was able to compare the glosses of words that are related to the words to be disambiguated.

4.5.2.3 Distributional Semantic Measures

Distributional similarity, as studied by language technology, covers an important kind of theories of word meaning and can hence be seen as contributing to semantic document indexing and retrieval. Its predecessors go back a long way, building on the notion of term dependence and structures derived therefrom (Luhn, 1957; Morris, Beghtol, and Hirst, 2003). Also called the contextual theory of meaning (see (Lyons, 1977) for the historical development of the concept), the underlying distributional hypothesis of (Harris, 1970) is often cited for explaining how word meaning enters information processing (Karlgren and Sahlgren, 2001; Sahlgren, 2006), and basically equals the claim "meaning is use" in language philosophy (Wittgenstein, 1967). Before attempts to utilize lexical resources for the same purpose, this used to be the sole source of word semantics in information retrieval, inherent in the exploitation of term occurrences (tfidf) and term co-occurrences (Wilks et al., 1990; Gallant, 1991; Gallant et al., 1992; Peat and Willett, 1991; Grefenstette, 1992; Schutze and Pedersen, 1997), including multiple-level term co-occurrences (Kontostathis, 2006; Kontostathis and Pottenger, 2006).

Lexical knowledge bases are expensive resources: creating one requires human experts, is time intensive and rather brittle to changes in language. Once created, updating the resource is again expensive and tends to lag between the current state of language usage/comprehension and the semantic network representing it (Mohammad and Hirst, 2005). On the other hand, large corpora may be collected by a simple web crawler. Large corpora of more formal writing are also available (for example, the Wall Street Journal or the American Printing House for the Blind (APHB) corpus) (Mohammad and Hirst, 2005). Therefore, using an appropriate distributional measure that best captures the semantic similarity-predicting information, may play a vital role in case of distributional measures (Mohammad and Hirst, 2005).

Distributional similarity is a corpus-dependent relation on terms (Budanitsky and Hirst, 2006). A major limitation of hand-generated lexicons is the relatively poor coverage of technical and scientific terms (Turney, 2001). Ontologies have been made for specific domains, which may be used to determine semantic similarity specific to these domains. However, the number of such ontologies is very limited. On the other hand, large amounts of corpora specific to particular domains are much easier to collect, allowing a widespread use of distributional domain-specific similarity (Mohammad and Hirst, 2005).

Given a text corpus, individual terms have more or less different contexts around them. Terms that occur within a certain window of a target terms are called the co-occurrences of the terms. The window size may be a few terms on either side, the complete sentence, a paragraph or the entire document. Consider the following sentence (Mohammad and Hirst, 2005): *the plane flew through a cloud*. If the window size is the complete sentence, *flew* co-occurs with *the*, *plane*, *through*, *a* and *cloud*. The set of terms that co-occur with a term constitute the context of the term.

Distributional measures use statistics acquired from large text corpora to determine how similar the contexts of two terms are. These measures are also used as proxies to measures of semantic similarity as terms found in similar contexts tend to be semantically similar. This is known as the distributional hypothesis (Harris, 1970) and such measures have traditionally been referred to as measures of distributional similarity.

Terms that are distributionally similar often represent semantically related concepts, if two terms have many co-occurring terms then similar things are being said about both of them. And conversely, if two terms are semantically similar then they are likely to be used in a similar fashion in text and thus end up with many common co-occurrences (Budanitsky and Hirst, 2006).

Apart from distributional similarity, there also exists distributional relatedness. The latter uses raw text and co-occurrence information to determine semantic relatedness between two terms. There is a difference between co-occurrence and collocation (Manning and Schütze, 1999): collocation refers to grammatically bound elements that occur in a particular order, and co-occurrence refers to a more general phenomenon of terms that are likely to be used in the same context. Hence one can distinguish between distributional similarity and distributional relatedness (Schutze and Pedersen, 1997; Mohammad and Hirst, 2005). The distributional hypothesis is generic enough to be the basis for both distributional similarity and distributional relatedness (Harris, 1970).

An important characteristic of any distributional measure is whether it is a measure of distributional similarity or more generally that of distributional relatedness. It should also be noted that a measure of distributional similarity will assign a high score for closely related but dissimilar terms belonging to the same thematic role. This is a known limitation of the current measures of distributional similarity (Mohammad and Hirst, 2005).

Correlation measures are typically used to measure attribute dependencies. Pearson's correlation coefficient is probably the most widely used measure for quantitative attributes (Nazareth, Soofi, and Zhao, 2007). The correlation coefficient $\mathrm{corr}(X, Y)$ between two random variables X and Y with expected values $E(X)$ and $E(Y)$ and standard deviations σ_X and σ_Y is defined as:

$$\mathrm{corr}(X, Y) = \frac{\mathrm{cov}(X, Y)}{\sigma_X \sigma_Y} = \frac{E((X - E(X))(Y - E(Y)))}{\sigma_X \sigma_Y},$$

where E is the expected value operator and cov stands for covariance. The correlation measure is not sensitive to nonlinear dependencies which do not manifest themselves in the covariance and can thus miss important features. This is in contrast to mutual information (MI) (Kraskov et al., 2005), thus MI can be a useful

way to measure interdependencies between features. MI is defined as

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p_1(x)\, p_2(y)} \right),$$

which is not a metric. However, using the joint entropy $H(X,Y) = -\sum_{x,y} p_{x,y} \log(p_{x,y})$, the expression

$$d(X,Y) = H(X,Y) - I(X;Y)$$

is a metric.

Extending the corpus to the entire web, the normalized Google distance is defined as (Cilibrasi and Vitanyi, 2007):

$$\mathrm{NGD}(x,y) = \frac{\max\{\log f(x), log f(y)\} - \log f(x,y)}{\log N - \min\{\log f(x), \log f(y)\}},$$

where $f(x)$ denotes the number of pages containing x, and $f(x,y)$denotes the number of pages containing both x and y, as reported by Google. The authors found that the distance corresponds well with WordNet-based categories. Computations, however, take a long time, since each pair needs a search on Google.

4.5.2.4 Composite Measures

There are certain advantages in measuring semantic association by combining a network structure with corpus statistics. The incorporation of a manually built knowledge base may complement the statistical approach where understanding of the text is impossible. The statistical model can take advantage of a conceptual space structured by a lexical resource (Jiang and Conrath, 1997).

Statistical techniques typically suffer from the sparse data problem: they perform poorly when the terms are relatively rare, due to the scarcity of data. Hybrid approaches attempt to address this problem by supplementing sparse data with information from a lexical database (Resnik, 1995; Jiang and Conrath, 1997). In a semantic network, to differentiate between the weights of edges connecting a

node and all its child nodes, one needs to consider the link strength of each specific child link. This is a situation in which corpus statistics can contribute. Ideally the method chosen should be both theoretically sound and computationally efficient (Jiang and Conrath, 1997).

Text classification benefited from topic grouping based on a combined measure of thesaurus, context, and co-occurrence-based semantic similarity (Liu and Chua, 2001). The thesaurus-based similarity is defined as:

$$R_L(f_1, f_2) = \left\{ \begin{array}{ll} 1 & \text{if } f_1 \text{ and } f_2 \text{ are in the same synset, or } f_1 = f_2 \\ 0.8 & \text{if } f_1 \text{ and } f_2 \text{ have antonym relation} \\ 0.5 & \text{if } f_1 \text{ and } f_2 \text{ have hypernym or meronym relation} \\ 0 & \text{others} \end{array} \right\},$$

where f_i is a feature (term).

The co-occurrence-based correlation is calculated as:

$$R_{CO}(f_1, f_2) = \frac{df(f_1 \wedge f_2)}{df(f_1 \vee f_2)},$$

where $df(f_1 \wedge f_2)$ is the number of documents containing both f_1 and f_2, while $df(f_1 \vee f_2)$ is the number of documents containing f_1 or f_2.

To derive the context-based correlation, a term's context is defined as the set of non-trivial terms near the term. A term is said to be near the term if their term distance is less than a given threshold, for instance, 5. The context is represented by a context vector $cv(f_j)$. To derive $cv(f_j)$, first all candidate context words of f_j are ranked by their density values:

$$\rho_{jk} = m_j(f_k)/n(f_j),$$

where $n(f_j)$ is the number of occurrence of f_j and $m_j(f_k)$ is the number of occurrences of f_k near f_j. Then the top ten terms are selected as the context of f_j. If two terms have a very high context similarity, it will have a high possibility that

they are semantically related. The context-based correlation is calculated between two terms as:

$$R_C(f_1, f_2) = \frac{\sum_{k=1}^{10} R_{CO}(f_{1k}, f_{2m(k)})\rho_{1k}\rho_{2m(k)}}{\sqrt{\sum_k \rho_{1k}^2}\sqrt{\sum_k \rho_{2k}^2}},$$

where $m(k) = \text{argmax}_s R_{CO}(f_{1k}, f_{2s})$.

Another composite measure was developed for word sense disambiguation. Following the notation in information theory, the information content (IC) of a concept/sense s can be quantified as follows.

$$\text{IC}(s) = \frac{1}{\log P(s)}.$$

where $P(s)$ is the probability of encountering an instance of sense s. In the case of the hierarchical structure, where a sense in the hierarchy subsumes those ones below it, this implies that $P(s)$ is monotonic as one moves up in the hierarchy. As the node's probability increases, its information content or its informativeness decreases. If there is a unique top node in the hierarchy, then its probability is 1, hence its information content is 0.

The Brown University Standard Corpus of Present-Day American English (Brown Corpus) is a general text collection that has been tagged by WordNet senses (Kucera and Francis, 1967). This corpus is frequently used as the basis for sense statistics. The corpus consists of 500 samples, distributed across 15 genres in rough proportion to the amount published in 1961 in each of those genres. All works sampled were published in 1961; as far as could be determined they were first published then, and were written by native speakers of American English. Each sample began at a random sentence-boundary in the article or other unit chosen, and continued up to the first sentence boundary after 2,000 words. The corpus originally (1961) contained 1,014,312 words sampled from 15 text categories:

1. PRESS: Reportage (44 texts)
    - Political
    - Sports
    - Society

- Spot News
- Financial
- Cultural

2. PRESS: Editorial (27 texts)
    - Institutional Daily
    - Personal
    - Letters to the Editor
3. PRESS: Reviews (17 texts)
    - theatre
    - books
    - music
    - dance
4. RELIGION (17 texts)
    - Books
    - Periodicals
    - Tracts
5. SKILL AND HOBBIES (36 texts)
    - Books
    - Periodicals
6. POPULAR LORE (48 texts)
    - Books
    - Periodicals
7. BELLES-LETTRES - Biography, Memoirs, etc. (75 texts)
    - Books
    - Periodicals

8. MISCELLANEOUS: US Government & House Organs (30 texts)
    - Government Documents
    - Foundation Reports
    - Industry Reports
    - College Catalog
    - Industry House organ
9. LEARNED (80 texts)
    - Natural Sciences
    - Medicine
    - Mathematics
    - Social and Behavioral Sciences
    - Political Science, Law, Education
    - Humanities
    - Technology and Engineering
10. FICTION: General (29 texts)
    - Novels
    - Short Stories
11. FICTION: Mystery and Detective Fiction (24 texts)
    - Novels
    - Short Stories
12. FICTION: Science (6 texts)
    - Novels
    - Short Stories
13. FICTION: Adventure and Western (29 texts)
    - Novels

- Short Stories

14. FICTION: Romance and Love Story (29 texts)
    - Novels
    - Short Stories
15. HUMOR (9 texts)
    - Novels
    - Essays, etc.

Even for quite large samples such as the Brown corpus, graphing words in order of decreasing frequency of occurrence shows a hyperbole: the frequency of the n-th most frequent word is roughly proportional to $1/n$. Thus "the" constitutes nearly 7% of the Brown Corpus, "to" and "of" more than another 3% each; while about half the total vocabulary are hapax legomena: words that occur only once in the corpus. This simple rank-vs.-frequency relationship is known as Zipf's law (Zipf, 1935; Zipf, 1949). To calculate the information content of a concept, each noun that occurred in the corpus was counted as an occurrence of each taxonomic class of WordNet containing it (plural nouns counted as instances of their singular forms) (Resnik, 1995). For example, an occurrence of *ship* would be counted toward the frequency of its hypernyms *vessel*, *craft*, *vehicle*, and so forth.

The information content increases as senses gets more and more abstract (that is, they are higher in the WordNet hypernym hierarchy, see Figure 4.2).

Given the monotonic feature of the information content value, the similarity of two senses can be formally defined as follows.

$$\mathrm{sim}(\mathrm{s}_1, \mathrm{s}_2) = \max_{s \in \mathrm{Sup}(s_1, s_2)} \mathrm{IC}(s) = \max_{s \in \mathrm{Sup}(s_1, s_2)} -\log p(s)$$

where $\mathrm{Sup}(s_1, s_2)$ is the set of senses that subsume both s_1 and s_2. To maximize the representativeness, the similarity value is the information content value of the node whose IC value is the largest among those higher order classes.

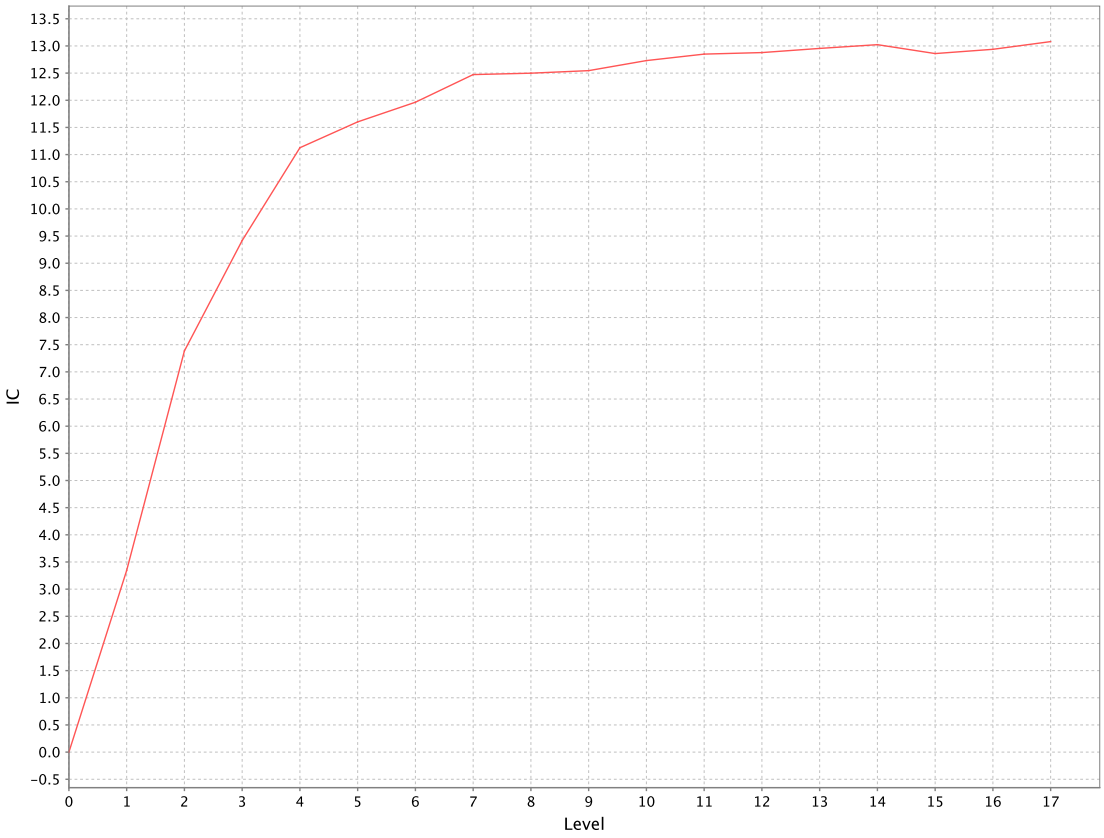


Figure 4.2: Average information content of senses at different levels of the WordNet hypernym hierarchy (logarithmic scale)

In the case of multiple inheritances, where terms can have more than one sense and hence multiple direct higher order classes, term similarity can be determined by the best similarity value among all the class pairs to which their various senses belong:

$$sim(f_1, f_2) = \max_{s_1 \in sen(f_1), s_2 \in sen(f_2)} sim(s_1, s_2)$$

where $sen(t)$ denotes the set of possible senses for term t.

There are some slightly different approaches toward calculating the concept probabilities in a corpus. Let us define two sense sets: terms(c) and classes(t). Terms(s) is the set of terms subsumed by the sense s. This can be seen as a sub-

tree in the whole hierarchy, including the sub-tree root s. Classes(w) is defined as the senses in which the term t is contained:

$$\text{classes}(t) = \{s | t \in terms(s)\}$$

(Resnik, 1995) defined a simple class/concept frequency formula:

$$\text{freq}(s) = \sum_{t \in terms(s)} freq(t).$$

(Richardson and Smeaton, 1995) proposed a slightly different calculation by considering the number of term senses factor:

$$\text{freq}(s) = \sum_{t \in terms(s)} \frac{freq(t)}{|senses(w)|}$$

Finally, the class/concept probability can be computed using maximum likelihood estimation (Jiang and Conrath, 1997):

$$p(s) = \frac{freq(s)}{N}$$

The information content method requires less information on the detailed structure of a lexical resource and it is insensitive to varying link types (Resnik, 1995). On the other hand, it does not differentiate between the similarity values of any pair of senses in a sub-hierarchy as long as their lowest super-ordinate sense is the same. Moreover, in the calculation of information content, a polysemous term will have a large content value if only term frequency data are used (Richardson and Smeaton, 1995).

Consider the link strength factor. The strength of a child link is proportional to the conditional probability of encountering an instance of the child sense s_i given an instance of its parent sense $p : P(s_i|p)$.

$$P(s_i|p) = \frac{P(s_i \cap p)}{P(p)} = \frac{P(s_i)}{P(p)}$$

Notice that the definition and determination of the information content indicate that s_i is a subset of p when a sense's informativeness is concerned. Define the link strength (LS) by taking the negative logarithm of the above probability:

$$\mathrm{LS}(s_i, p) = -\log(P(s_i|p) = IC(s_i) - \mathrm{IC}(p)$$

This states that the link strength (LS) is simply the difference of the information content values between a child concept and its parent concept.

Considering other factors, such as local density, node depth, and link type, the overall edge weight (wt) for a child node s and its parent node p can be determined as follows:

$$\mathrm{wt}(s, p) = \left(\beta + (1-\beta)\frac{\bar{E}}{E(p)}\right)(\frac{d(p)+1}{d(p)})^{\alpha}[\mathrm{IC}(s) - \mathrm{IC}(p)]T(s, p),$$

where $d(p)$ denotes the depth of the node p in the hierarchy, $E(p)$ the number of edges in the child links (i.e. local density), $\bar{E}$ the average density in the whole hierarchy, and $T(s, p)$ the link relation/type factor. The parameters α ($\alpha \geq 0$) and β ($0 \leq \beta \leq 1$) control the degree of how much the node depth and density factors contribute to the edge weighting computation. For instance, these contributions become less significant when α approaches 0 and β approaches 1. The overall distance between two nodes would thus be the summation of edge weights along the shortest path linking two nodes (Jiang and Conrath, 1997).

$$d(s_1, s_2) = \sum_{s \in \{\mathrm{path}(s_1, s_2) - \mathrm{LSuper}(s_1, s_2)\}} \mathrm{wt}(s, \mathrm{parent}(s))$$

where $s_1 \in \mathrm{sen}(f_1)$, $s_2 \in \mathrm{sen}(f_2)$, and $\mathrm{path}(s_1, s_2)$ is the set that contains all the nodes in the shortest path from s_1 to s_2. One of the elements of the set is $\mathrm{LSuper}(s_1, s_2)$, which denotes the lowest super-ordinate of s_1 and s_2. In the special case when only link strength is considered in the weighting scheme of the above equation, i.e. $\alpha = 0$, $\beta = 1$, and $T(s, p) = 1$, the distance function can be simplified

as follows:

$$d(s_1, s_2) = \mathrm{IC}(s_1) + \mathrm{IC}(s_2) - 2\mathrm{IC}(\mathrm{LSuper}(s_1, s_2))$$

This distance measure also satisfies the properties of a metric (Jiang and Conrath, 1997).

In an experiment, Jiang and Conrath's metric did just a little worse using Roget's Thesaurus than the results using WordNet (McHale, 1998), the authors claimed that Roget's Thesaurus captured the popular similarity of isolated term pairs better than WordNet.

4.6 Methodology for Text Classification

In testing a method for text categorization it is important that knowledge of the nature of the test data does not unduly influence the development of the system, or the performance obtained will be unrealistically high. One way of dealing with this is to divide a set of data into two subsets: a training set and a test set. A categorization system is developed by automated training on the training set only, and/or by human knowledge engineering based on examination of the training set only. The categorization system is then tested on the previously unexamined test set.

4.6.1 Performance Measures

Effectiveness results can only be compared between studies that are tested on the same training and test sets. However, as researchers tend to use test benchmark collections slightly differently, such comparisons should be handled with care (Sebastiani, 2002).

Text classification systems are normally evaluated by measures of effectiveness rather than efficiency, that is, their ability to take the right classification

decisions.

Category ranking can be evaluated using measures similar to the conventional measures for evaluating ranking-based document retrieval systems: recall, precision, and 11-point average precision. Given a classifier whose input is a document x_j, and whose output is a ranked list of categories assigned to that document, the recall and precision can be computed at any threshold on this ranked list:

$$\text{precision}(x_j) = \frac{\text{categories found and correct}}{\text{total categories found}},$$

$$\text{recall}(x_j) = \frac{\text{categories found and correct}}{\text{total categories correct}},$$

where "categories found" means categories above the decision threshold. For the global evaluation of a classifier on a collection of test documents, one may adapt the procedure for the conventional interpolated 11-point average precision (Salton and McGill, 1983), as described below (Yang and Liu, 1999):

1. For each document, compute the recall and precision at each position in the ranked list where a correct category is found.

2. For each interval between recall thresholds of 0 %, 10 %, 20 %,..., 100 %, use the highest precision value in that interval as the representative precision value at the left boundary of this interval.

3. For the recall threshold of 100 %, the representative precision is either the exact precision value if such a data point exists, or the precision value at the closest point in terms of recall. If the interval is empty, use the default precision value of zero.

4. Interpolation: At each of the above recall thresholds, replace the representative precision using the highest score among the representative precision values at this threshold and the higher thresholds.

A different approach is to calculate precision and recall with regard to a class c_k. Estimates can be obtained as (Sebastiani, 2002):

$$\text{precision}(c_k) = \frac{TP_k}{TP_k + FP_k},$$

$$\text{recall}(c_k) = \frac{TP_k}{TP_K + FN_k},$$

where TP_k is the number of correctly classified instances under c_k, FP_k is the number of false positives, and FN_k is the number of false negatives, that is, the errors of omission (see Section 3.7).

To obtain overall estimates of precision and recall in both approaches, two different methods can be used:

- micro-averaged: where precision and recall are obtained by summing over all individual decisions;
- macro-averaged: where precision and recall are first evaluated "locally" for each category, and then "globally" by averaging over the results of the different categories.

These two methods may give quite different results, especially if the different categories have very different generality. The ability of a classifier to behave well also on categories with low generality (that is, categories with few positive training instances) will be emphasized by macro-averaged and much less so by micro-averaged methods. Therefore the two methods will be equalized only on the uniform category distribution data sets. Whether one or the other should be used obviously depends on the actual application requirements (Sebastiani, 2002).

Precision and recall should be interpreted together, they are not sensible measures of effectiveness in themselves. It is well known from the information retrieval practice that higher levels of precision may be obtained at the price of low values of recall (van Rijsbergen, 1979). A classifier should thus be evaluated by

means of a measure which combines precision and recall. To accomplish this, several measures have been proposed. Among them, the two most widely used measures adopted by text classification are the F_β function and the breakeven point.

The F_β ($0 \leq \beta < \infty$) measure is a composite measure of precision and recall (van Rijsbergen, 1979; Lewis, 1995; Moulinier, Raškinis, and Ganascia, 1996):

$$F_\beta = \frac{(\beta^2 + 1)pr}{\beta^2 p + r}.$$

β can be seen as the relative importance of precision and recall. With $\beta = 0$ F_β equals precision, while if $\beta \to \infty$ then F_β coincides with recall. Setting $\beta = 1.0$, F_β corresponds to the harmonic mean of recall and precision:

$$F_1 = \frac{2pr}{p + r}.$$

The breakeven point is the value at which precision equals recall (Lewis, 1992; Apté, Damerau, and Weiss, 1994a; Dagan, Karov, and Roth, 1997; Joachims, 1998). To obtain the breakeven point, a plot of precision as a function of recall is computed similarly to the 11-point plot; the breakeven point value is the value of precision or recall for which the plot intersects the precision = recall line. Breakeven may not be a good effectiveness measure (Sebastiani, 2002), as

- There may be no parameter setting that yields the breakeven; in this case the final breakeven value, obtained by interpolation, is artificial;
- To have recall equal precision is not necessarily desirable, and it is not clear that a system that achieves high breakeven can be tuned to score high on other effectiveness measures.

It was also noted that when for no value of the parameters precision and recall are close enough, interpolated breakeven may not be a reliable indicator of effectiveness (Yang and Liu, 1999).

Compared with the F_1 function, (Yang and Liu, 1999) showed that the breakeven point of a classifier is always less or equal than its F_1 value.

Although accuracy is commonly used in the machine learning literature, it is not widely used in TC. The large value of documents in the whole corpus makes them much more insensitive to variations in the number of correct decisions than precision and recall (Yang and Liu, 1999). This makes the classifier behave like a trivial rejector.

4.6.2 Benchmark Text Collections

The most widely used benchmark corpus is the Reuters collection. The documents in the Reuters collection appeared on the Reuters newswire in 1987. The documents were assembled and indexed with categories by personnel from Reuters Ltd. and Carnegie Group, Inc. in 1987. The older versions of the Reuters collection (namely Reuters-22173 and Reuters-21450) account for a good deal of the experimental work in text categorization (Hayes and Weinstein, 1990; Lewis and Ringuette, 1994; Apté, Damerau, and Weiss, 1994b; Wiener, Pedersen, and Weigend, 1995; Cohen and Singer, 1996; Ng, Goh, and Low, 1997; Yang, 1999), however, the original release had many inconsistencies. The collection was cleaned up, the new collection has only 21,578 documents, and thus is called the Reuters-21578 collection (Lewis, 1999). Conducting experiments on this popular corpus provides a sensible comparison with results published by other authors.

For benchmarking purposes, the ModApte split was adapted. This split assigns documents from April 7, 1987 and before to the training set, and documents from April 8, 1987 and after to the test set.

As pointed out in the manual of the collection (Lewis, 1999), testing a system only on the "easy" categories is a mistake often committed by researchers. Some of the 135 categories have few or no positive training examples or few or no positive

Training Set	9603 documents
Test Set	3299
Unused	8676

Table 4.2: Number of training and test documents

test examples or both. Purely supervised learning systems do very badly on these categories. Knowledge-based systems, on the other hand, tend to do well on them. These comparisons are of no interest in this thesis, hence only those ninety categories which have at least one positive example were included in the benchmark. The number of categories per document is 1.3 on average. The category distribution is skewed, the most common category has a training-set frequency of 2877, but 82 % of the categories have less than 100 instances, and 33 % of the categories have less than 10 instances.

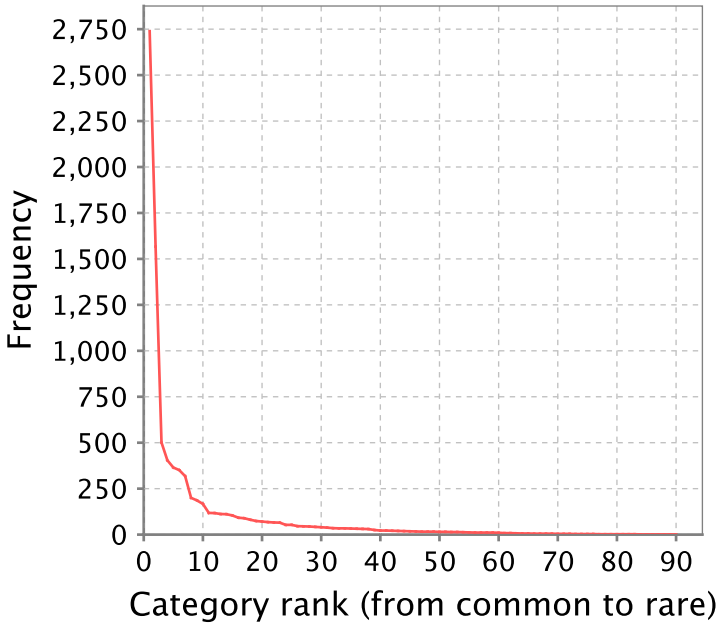


Figure 4.3: Class frequencies in the training set

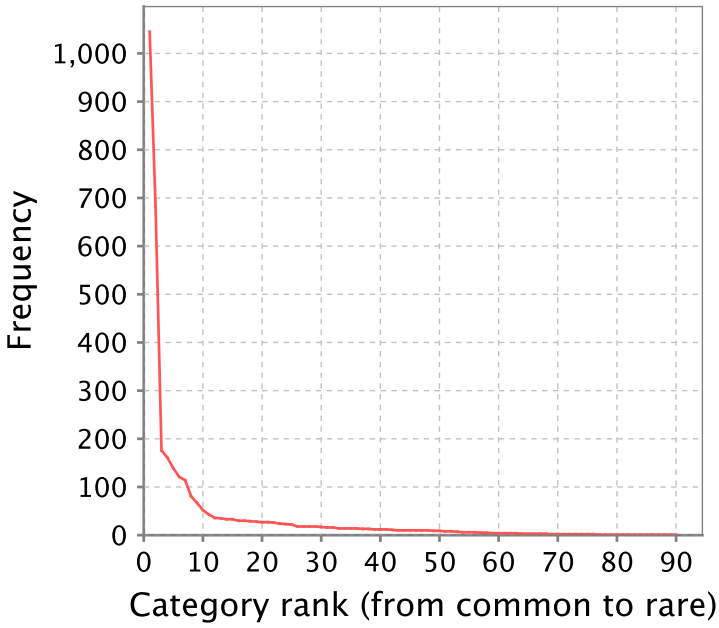


Figure 4.4: Class frequencies in the test set

Another benchmark data corpus is the 20 Newsgroups corpus, which is a collection of approximate 20,000 newsgroup documents nearly evenly divided among 20 discussion groups and each document is labeled as one of the 20 categories corresponding to the name of the newsgroup that the document was posted to. Some newsgroups are very closely related to each other. For example, the posts in category of comp.sys.ibm.pc.hardware are very similar to those in category of comp.sys.mac.hardware. However, others are highly unrelated, for example, the category of misc.forsale and category of soc.religion.christian. After removing duplicates and headers, the remaining 18846 documents are sorted by date and are partitioned into 11314 training documents (about 60%) and 7532 test documents (about 40%). After this partition, the training and test documents still remain nearly evenly distribution on the 20 topics. Therefore, compared with the skewed category distribution in the Reuters corpus, the 20 categories in the 20 Newsgroups

corpus are of approximately uniform distribution. Three different splits were made to benchmark the sensitivity of kernels to the amount of data available for training, using 50 %, 60 %, and 70 % of the collection as the training set.

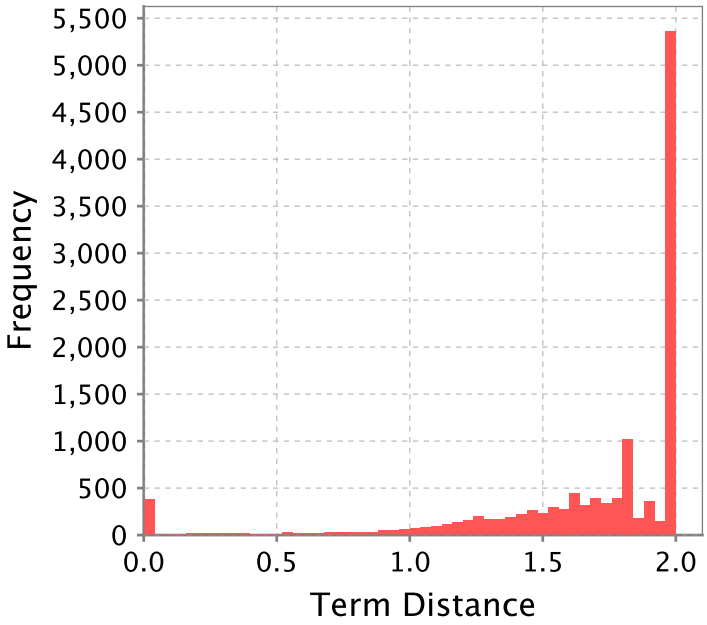


Figure 4.5: Distribution of distances between adjacent terms in alphabetic order

4.7 Experimental Results

Similar to the benchmarks in Chapter 3, first we discuss the quality and importance of reordering the features 4.7.1. Results based on the framework presented in the previous section are reported in Section 4.7.2. These results are important when comparing with other algorithms in the scientific literature. We also find it important to see how our method fares in an actual application, so we conducted experiments in a digital library scenario 4.7.3.

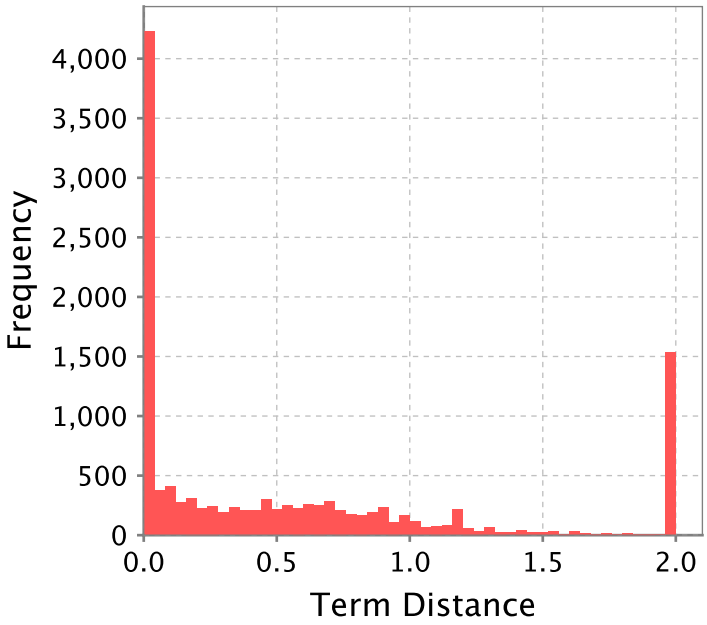


Figure 4.6: Distribution of distances between adjacent terms in a semantic order based on Jiang-Conrath distance

4.7.1 The Importance of Ordering

Prior to the semantic ordering, terms are assumed to be in alphabetic order, though any arbitrary order could be assumed. A slight modification of Equation 4.2 leads to a more conservative term similarity approximation. Whereas Equation 4.2 takes the maximum relatedness of the senses of two terms as the similarity between the terms, taking the minimum will result in an order where only closely related terms are adjacent:

$$\mathrm{rel}(f_1, f_2) = \min_{s_1 \in \mathrm{sen}(f_1), s_2 \in \mathrm{sen}(f_2)} \mathrm{rel}(s_1, s_2). \tag{4.3}$$

Measuring the Jiang-Conrath distance between adjacent terms, the average distance is 1.68215255170455 on the vocabulary of Reuters-21578. Note that the Jiang-Conrath distance is normalized to the interval $[0, 2]$. Figure 4.5 shows the distribution of distances. The histogram has a high peak at the maximum distance,

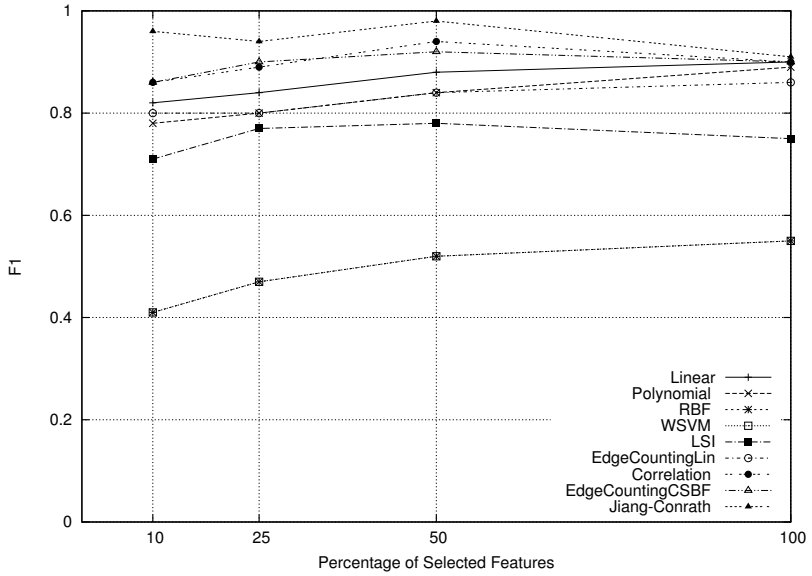


Figure 4.7: Micro-average F1 versus percentage of features, Reuters data set, Top-10 categories

indicating that the alphabetic order has little to do with semantic distance. However, there are few terms with zero or little distance between them. This is due to terms which are related and start with the same word or stem. For example, *account, account executive, account for, accountable, accountant, accounting principle, accounting standard, accounting system, accounts payable, accounts receivable.*

The same average distance after reordering the terms with the proposed algorithm and the Jiang-Conrath distance is 0.5646989056177. About one third of the terms has very little distance between each other (see Figure 4.6). Nevertheless, over 10 % of the total terms still has the maximum distance. This is due to the non-optimal nature of the proposed term-ordering algorithm. These terms add noise to the classification. The noisy terms occur typically at the two sides of the scale, that is, the leftmost terms and the rightmost terms. While it is easy to find close terms in the beginning, as the algorithm proceeds, fewer terms remain

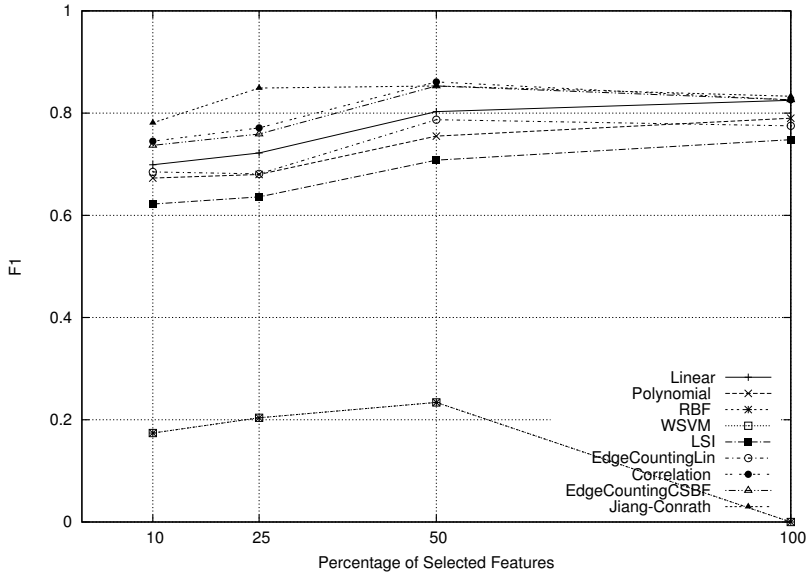


Figure 4.8: Macro-average F1 versus percentage of features, Reuters data set, Top-10 categories

in the pool to be chosen. For instance, *brand, brand name, trade name, label* are in the 33rd, 34th, 35th and 36th position on the left side counting from the seed respectively, while *windy, widespread, willingly, whatsoever, worried, worthwhile* close the left side, apparently sharing little in common. The noise can be reduced by the appropriate choice of length of the support of the basis functions (see Section 3.2), so the impact of adjacent, but distantly related terms can be minimized. For more on the importance and quality of reordering, see Section 3.8.1.

4.7.2 Results on Benchmark Text Collections

The benchmarks compare traditional kernels (linear, polynomial, RBF), a wavelet kernel (Zhang, Zhou, and Jiao, 2004), linear smoothing kernels (with distributional patterns (LSI) and lexical resource-based semantic matrix (edge-counting)), and CSBF kernels with three types of distances: distributional (correlation), lexical

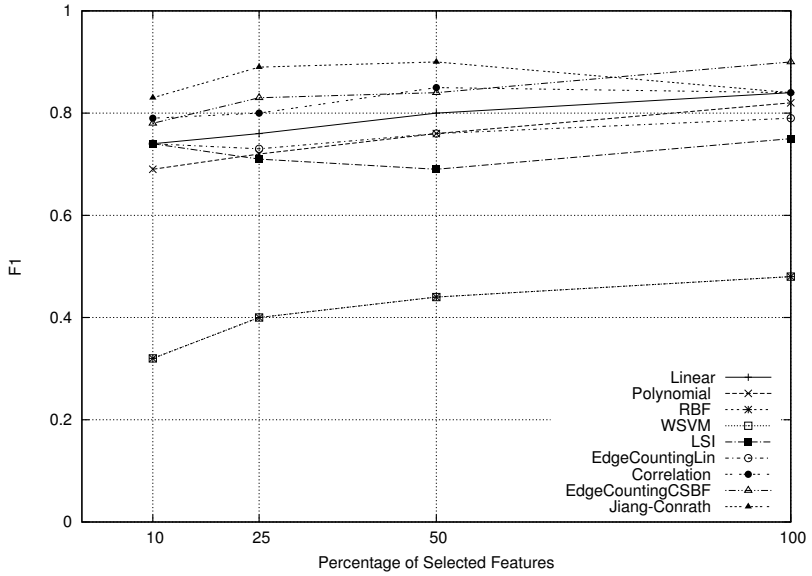


Figure 4.9: Micro-average F1 versus percentage of features, Reuters data set, all categories

resource-based (edge-counting), and composite measure (Jiang-Conrath). With regards to Sections 4.4 and 4.5.2.2, this set up allows us to test the state-of-the-art, and it also enables us to compare the efficiency of distributional and lexical resource-based distances in general. We also performed feature selection with the same method as in Section 3.8.2 to understand whether keeping all features would be beneficial to the overall performance.

In preparing the index terms, the vocabulary is restricted to the terms of WordNet 3.0 in order to be able to calculate the similarity score between any two terms. Stop words were removed in advance. Multiple word expressions were used to fully utilize WordNet. We used the built-in stemmer of WordNet, which is able to distinguish between different parts-of-speeches if the form of the word is unambiguous. For example, {accommodates, accommodated, accommodation} was stemmed to {accommodate, accommodate, accommodation}. We used term

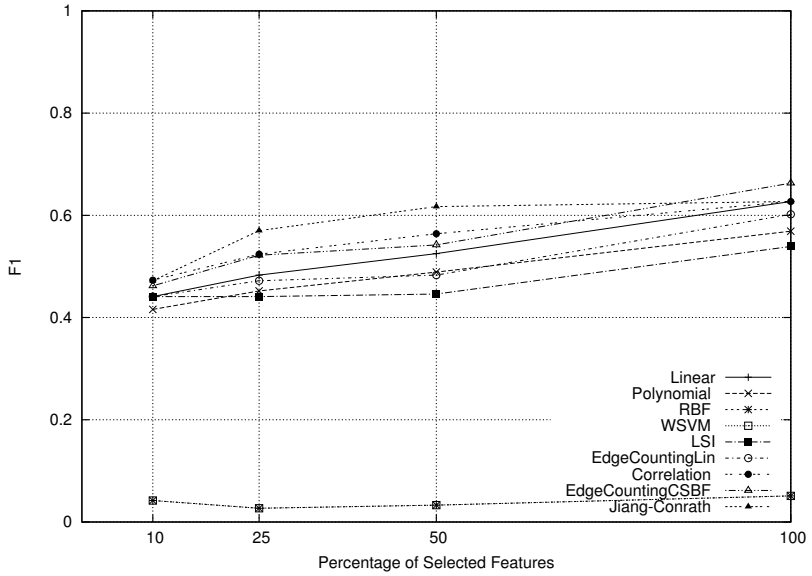


Figure 4.10: Macro-average F1 versus percentage of features, Reuters data set, all categories

frequency as term weighting, and the same controlled vocabulary for purely distributional models to achieve better comparability.

We applied the `libsvm` (Chang and Lin, 2001) library to benchmark the baseline kernels, and implement the suggested kernel. We used only C-SVMs, for each kernel a wide range of kernel-specific parameters were benchmarked. Figures 4.7, 4.8, 4.9, 4.10, 4.11, 4.12, 4.13, 4.14, 4.15 and 4.16 summarize the results for the baseline kernels and CSBF kernels with the best parameter settings for each kernel. Linear kernel refers to a simple linear kernel without semantic smoothing. The latent semantic kernel was tested with keeping 300 dimensions, as this setting gave the best results in (Cristianini, Shawe-Taylor, and Lodhi, 2002); our results confirm that the latent semantic kernel is able to closely approximate the performance of, but not to outperform, a linear kernel. Linear edge counting refers to the semantic smoothing kernel defined by (Siolas and d'Alché Buc, 2000). It should be noted that

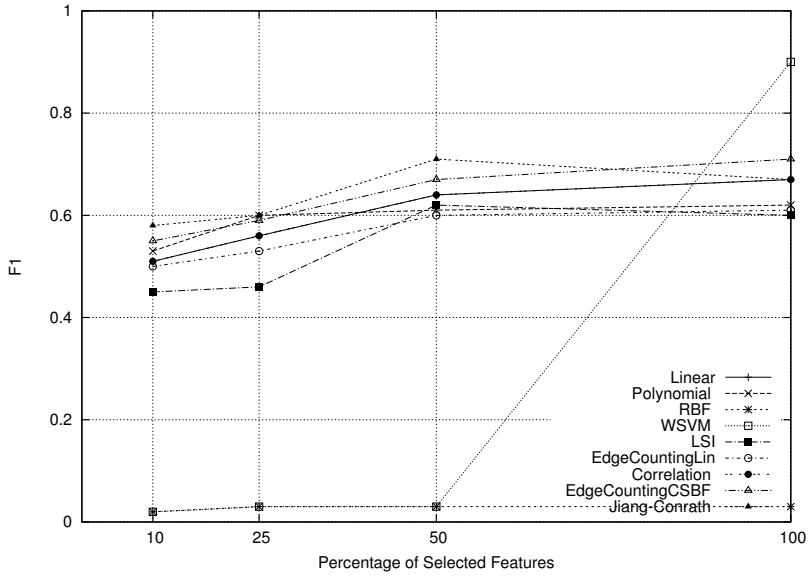


Figure 4.11: Micro-average F1 versus percentage of features, 20News 50 % training data

this semantic smoothing kernel improves performance if the training set is smaller, or there are categories with few training instances (20News 50 % and Reuters All).

The linear kernel is parameter-free, hence only one run was performed for each data set. For polynomial kernels, the degree was varied (2 and 3), as well as the offset (0 and 1). RBF kernels converge very slowly in the `libsvm` implementation, therefore only the default parameter value (one over the number of features) and unit γ were benchmarked on all data sets, while a wider range of settings showed the insensitivity of the parameters on smaller collections. WSVM kernels were tested with parameters 1, 2.5, 5 and 10; however, this kernel also proved insensitive to the choice of parameters. All kernels are reported with a result for the best parameter setting.

When benchmarking CSBF kernels on binary classification problems in the previous chapter, we observed that the difference between equally spaced observa-

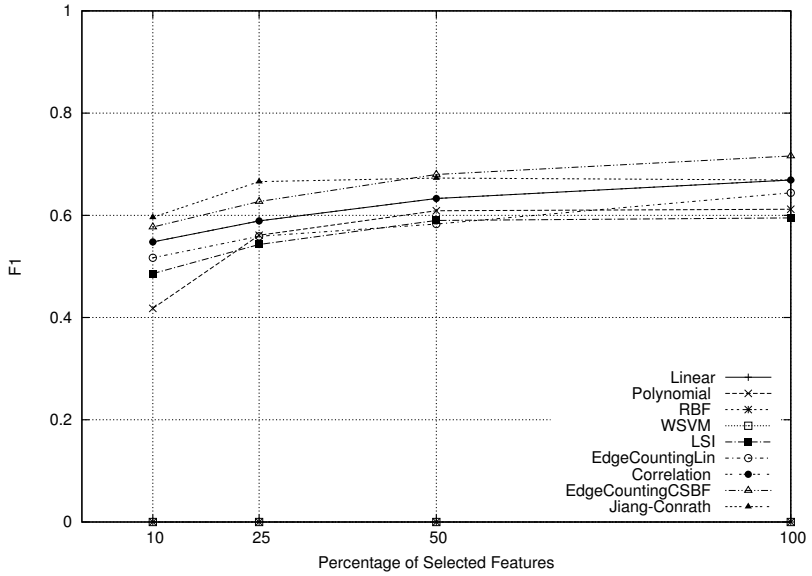


Figure 4.12: Macro-average F1 versus percentage of features, 20News 50 % training data

tions and randomly spaced observations is not significant, or worse in the second case. Therefore we did not benchmark randomly spaced observations for textual data.

For correlation measure, the optimal support was of length 1 (or 0.5), that is, the equivalent of a linear kernel (see Appendix for detailed data). Polysemy seems to introduce too much noise, statistical relatedness is established on the training set, and does not necessarily hold for the test set if the terms appear in different senses.

Edge-counting, as a purely lexical resource-based distance metric, fares much better (see Appendix for detailed data). Note that we use Equation 4.3 instead of Equation 4.2, thus making sure that the ordering algorithm places only terms next to each other that are closely related. As it was the case with the edge-counting-based linear semantic smoothing kernels, the greatest improvements can be seen if

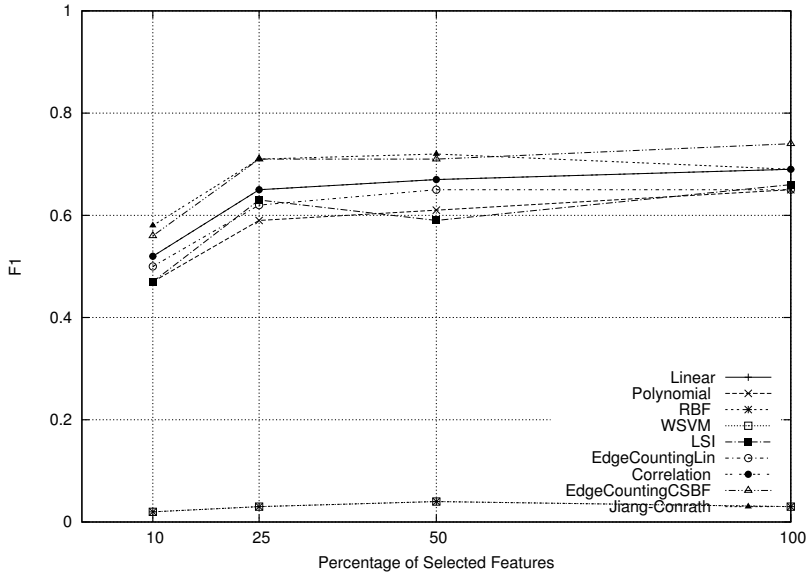


Figure 4.13: Micro-average F1 versus percentage of features, 20News 60 % training data

the training set was relatively small, or there were only a few training instances for some of the classes. Performance peaked with a support length of 5 or 10.

By far the best results were obtained with a Jiang-Conrath distance based wavelet kernel, with support length 5 (see Appendix for detailed data). This is a composite measure incorporating distributional information, as well as information of a lexical hierarchy. The only exception being the Reuters database with all categories, where this kernel underperforms most other kernels. Unfortunately, this was the kernel that produced the most variance in the results, and without parameter tuning, this kernel might not be optimal.

Irrespective of the collection, it should be obvious that keeping all features results in higher performance, especially with the proposed kernels. The only exceptions is the RBF kernel, which drops in performance as the number of kept features increases.

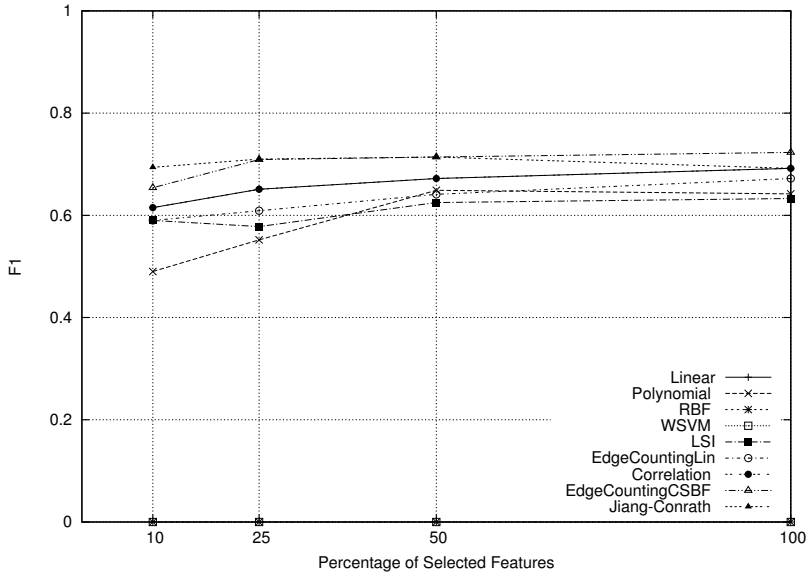


Figure 4.14: Macro-average F1 versus percentage of features, 20News 60 % training data

When comparing with other state-of-the-art text classification methods, the proposed kernels are on a par with them. A recent paper introducing a supervised term weighting method reports a maximum micro-averaged F_1 value of 0.93 for Reuters-21578 top ten categories with support vector machines (Lan et al., 2009). CSBF kernels certainly outperform this value when incorporating prior knowledge. The same paper reports on the 20News data set with a 60% split. These results are consistently higher even for linear kernels with simple term frequency weighting. We believe it is due to the difference in indexing: (Lan et al., 2009) uses algorithmic stemming, while we used a controlled vocabulary.

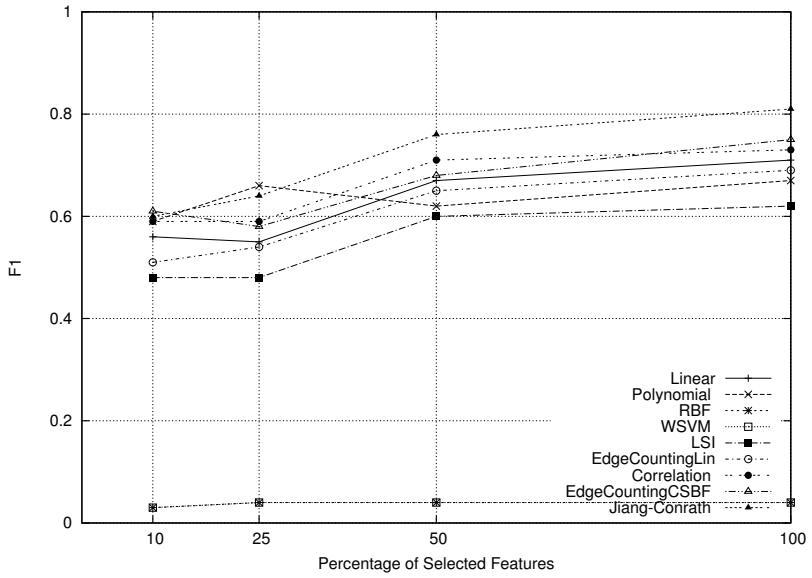


Figure 4.15: Micro-average F1 versus percentage of features, 20News 70 % training data

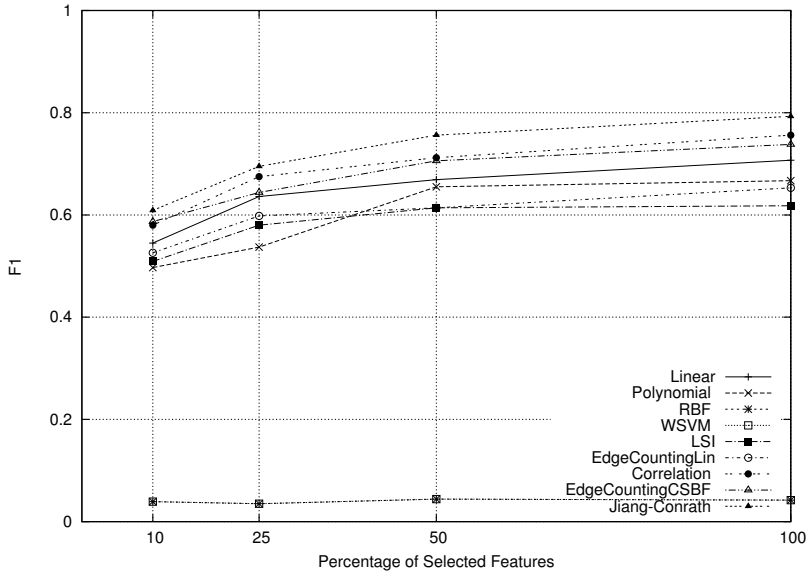


Figure 4.16: Macro-average F1 versus percentage of features, 20News 70 % training data

Table 4.3:

	Linear	Polynomial	RBF
Microaverage P	0.744	0.722	**0.880**
Macroaverage P	0.685	0.617	**0.976**
Microaverage R	**0.686**	0.623	0.017
Macroaverage R	**0.464**	0.398	0.064
Microaverage F_1	**0.714**	0.669	0.033
Macroaverage F_1	**0.553**	0.484	0.121

Table 4.4: Results on abstracts with traditional kernels, top-level categories

	Linear	Polynomial	RBF
Microaverage P	0.744	0.722	**0.880**
Macroaverage P	0.685	0.617	**0.976**
Microaverage R	**0.686**	0.623	0.017
Macroaverage R	**0.464**	0.398	0.064
Microaverage F_1	**0.714**	0.669	0.033
Macroaverage F_1	**0.553**	0.484	0.121

Table 4.5: Results on abstracts with traditional kernels, refined categories

	Linear	Polynomial	RBF
Microaverage P	0.514	0.457	**0.680**
Macroaverage P	0.603	0.595	**0.951**
Microaverage R	**0.433**	0.348	0.012
Macroaverage R	**0.364**	0.295	0.174
Microaverage F_1	**0.470**	0.395	0.023
Macroaverage F_1	**0.454**	0.395	0.294

4.7.3 An Application in Digital Libraries

We found in the previous section that the proposed kernels work well in sparse text classification problem, and we wanted to see how it compares in an actual application. We chose a digital library with a large number of keywords: the computational feasibility of linear semantic kernels depends on the number of index terms, and not the size of the database. We chose Strathprints as the text collection, since it has a fairly small size, but has a large number of index terms.

Strathprints is "an institutional eprint repository for making research papers and other scholarly publications widely available on the Internet" at the University of Strathclyde, UK (Dawson and Slevin, 2008), its hosting and technical support provided by the Department of Computer and Information Sciences (CIS). Eprints and usage statistics software have been installed, configured and managed by the

Table 4.6: Results on abstracts with L_2 kernels, top-level categories

Support length	2	4	6	8	10
Microaverage P	**0.732**	0.713	0.704	0.696	0.695
Macroaverage P	**0.685**	0.660	0.657	0.647	0.642
Microaverage R	**0.680**	0.672	0.671	0.668	0.666
Macroaverage R	**0.469**	0.462	0.467	0.462	0.459
Microaverage F_1	**0.705**	0.692	0.687	0.682	0.680
Macroaverage F_1	**0.557**	0.544	0.546	0.539	0.536

Table 4.7: Results on abstracts with L_2 kernels, refined categories

Support length	2	4	6	8	10
Microaverage P	**0.516**	0.503	0.485	0.472	0.466
Macroaverage P	**0.611**	0.595	0.572	0.558	0.547
Microaverage R	**0.444**	0.441	0.439	0.435	0.433
Macroaverage R	**0.362**	0.360	0.361	0.357	0.358
Microaverage F_1	**0.478**	0.470	0.461	0.453	0.449
Macroaverage F_1	**0.455**	0.449	0.442	0.435	0.432

Table 4.8: Results on full texts with traditional kernels, top-level categories

	Linear	Polynomial	RBF
Microaverage P	0.728	0.591	**0.949**
Macroaverage P	0.555	0.302	**0.992**
Microaverage R	**0.738**	0.694	0.568
Macroaverage R	**0.612**	0.564	0.289
Microaverage F_1	**0.733**	0.638	0.711
Macroaverage F_1	**0.582**	0.393	0.448

Table 4.9: Results on full texts with traditional kernels, refined categories

	Linear	Polynomial	RBF
Microaverage P	0.487	0.335	**0.880**
Macroaverage P	0.571	0.366	**0.980**
Microaverage R	**0.523**	0.514	0.230
Macroaverage R	0.576	**0.586**	0.440
Microaverage F_1	**0.505**	0.406	0.370
Macroaverage F_1	0.573	0.451	**0.610**

Table 4.10: Results on full texts with L_2 kernels, top-level categories

Support length	2	4	6	8	10
Microaverage P	0.714	0.724	0.**728**	0.712	0.695
Macroaverage P	0.490	0.482	**0.538**	0.533	0.532
Microaverage R	**0.738**	0.731	0.728	0.713	0.666
Macroaverage R	**0.612**	0.588	0.584	0.574	0.459
Microaverage F_1	0.726	0.728	**0.728**	0.712	0.680
Macroaverage F_1	0.545	0.530	**0.560**	0.553	0.496

Table 4.11: Results on full texts with L_2 kernels, refined categories

Support length	2	4	6	8	10
Microaverage P	0.462	0.474	**0.488**	0.480	0.463
Macroaverage P	0.556	0.558	**0.573**	0.569	0.557
Microaverage R	**0.505**	0.501	0.493	0.485	0.478
Macroaverage R	**0.561**	0.499	0.482	0.473	0.463
Microaverage F_1	0.482	0.488	**0.491**	0.483	0.471
Macroaverage F_1	**0.558**	0.523	0.523	0.521	0.510

Centre for Digital Library Research (CDLR) at the same university. Its digital objects are indexed by the LCSH classification scheme. From an ML point of view, this is a multilabel scenario where an instance of the collection may belong to several categories. Out of 6869 records, the size of the database on 13th June 2009, we downloaded and processed 5946 abstracts, the rest being records without abstracts or duplicates. With 14 ppt files removed, only abstracts in doc, html and pdf format were indexed by their LCSH tags. Keywords were obtained by a WordNet-based stemmer using the controlled vocabulary of the lexical database resulting in 21718 keywords in the full-text documents and 11586 in the abstracts. Keywords were ranked according to the Jiang-Conrath distance.

With 20 top classes and altogether 176 classes, the immediate research question was how efficiently SVM kernels can reproduce different levels of increasingly fine-grained text categories based on fulltext vs. abstracts only. The corpus was split to 80% training data and 20% test data; validation was not applied.

We split the multilabel, multiclass classification problems into one-against-all binary problems and calculated the micro-, and macro-averaged precision and recall values, and then their average, the F_1 score (Yang, 1999). For both sets of measurements, this was the most important observation parameter. Only C-SVMs were benchmarked, with the C penalty parameter left at the default value of 1. The implementation used the `libsvm` library (Chang and Lin, 2001).

We used the widespread linear, polynomial and RBF kernels on vectors to study classification performance. Polynomial kernels were benchmarked at second and third degree, RBF kernels were benchmarked with a small value ($\gamma = 1/$size of feature set) parameter as well as relatively high one ($\gamma = 1$ and 2). Because of the size of the fulltext-based vocabulary, the implementation of a linear semantic matrix was prohibitive.

A B-spline kernel with multiple parameters was benchmarked with the length

of support ranging between 2 and 10. In terms of the micro- and macroaverage F_1 measures, in three out of four cases the wavelet kernel outperformed the traditional kernels while reconstructing existing classification tags based on abstracts (Tables 4.4, 4.5, 4.8 and 4.9). In a repeated experiment based on fulltext documents, the opposite was the case, even if the difference in favour of traditional kernels was a minor one (Tables 4.6, 4.7, 4.10, and 4.11). We argue that this goes back to word sense disambiguation (Manning and Schütze, 1999): abstracts had less ambigous terms than complete documents. In all, the wavelet kernel performed best in the task of reconstructing the existing classification on a deeper level from abstracts.

The computational complexity of the proposed kernel is higher than that of linear, polynomial, and RBF kernels, but it is definitely lower than that of linear semantic kernels. When comparing actual running times, linear kernel training and testing finished in 307 seconds, polynomial in 381 seconds on average, RBF in 1322 seconds. The CSBF kernel with support length of 0.5 terminates in almost the same time as the linear kernel, in 318 seconds. However, increasing the support to 2 indicates that the overhead calculations are in fact considerable, since computations take slower than with an RBF kernels, terminating in 2673 seconds. Subsequent increases in the support length show a near linear scaling. While this result is discouraging, considering that the linear semantic kernels do not even scale to this magnitude of index terms, CSBF kernels are still a more viable alternative to incorporate prior knowledge in the classification process.

Chapter 5

Conclusion

Feature weighting is a powerful tool to discount less important features, but not fully eliminate them to maintain the richness of representation. Preserving the richness is particularly important when dealing with sparse data, such as text collections, or data sets with many, but highly correlated features. In fact, for sparse data, feature expansion is widely used, but not with weighted expansion features. This thesis offers a representation via kernel methods which expands the idea of feature expansion by weighting the expansion features. The proposed compactly supported basis function kernels draw on the concept of existing wavelet kernels, expanding on the core idea of applying wavelet analysis for support vector machines.

5.1 Contributions to Supervised Classification

Supervised machine learning, and support vector machines in particular, gain a powerful and computationally feasible method for feature engineering. While feature weighting and feature selection has been researched even in the kernel space prior to this work, the proposed combination of feature weighting and expansion is novel.

A more general framework for wavelet kernels has been introduced. While other wavelet kernels have been proved to be admissible, they did not use the inner product of the embedding L_2 space. By regarding the vector representation of an object as a sample of a hypothetical signal, a new family of wavelet kernels has been identified which uses the L_2 inner product. The underlying assumption is that there is a relation between consecutive features, this assumption has been addressed by an algorithm that orders the feature set according to an almost arbitrary relatedness measure.

The proposed kernels do not have additional storage needs, yet they are able to incorporate feature interdependence into calculating a similarity score. The running time and computational complexity of the kernels is close to that of a linear kernel, and can be adjusted by the length of the support of the basis function. By increasing the size of the cache of state-of-the-art implementations of support vector machines, the running time can be reduced significantly, though the number of iterations required will not decrease.

Experiments on a variety of general data set show significant improvement over baseline kernels. Especially if many features are relevant, but the features are strongly related to one another, the proposed kernels give a clear advantage.

The proposed model completely replaces existing representations retaining some beneficial properties of them: existing algorithms can easily be adopted to the new representation model. Kernel methods and support vector machines readily accept the new representation, hence the representation scales to data sets where these algorithms do.

5.2 Contributions to Text Representation

Text categorization and information retrieval, showed significant improvements with the proposed kernels. This result is in line with the findings on general data

sets: texts have many relevant and related features. The kernels seem to have a special edge over baseline methods if less training data is available.

The intricate nature term relatedness gave way to several semantic relatedness measures: both distributional measures and lexical resource-based measures, as well as combined approaches have been developed. These measures were not explored for use in large-scale experiments before, they were employed only in computationally demanding natural language processing tasks, such as word-sense disambiguation. CSBF kernels integrate these measures efficiently, hence they are able to capture prior knowledge that may be encoded in the relatedness measure.

If the wavelet kernels are regarded as a model of language representation, Two, seemingly contradicting linguistic theories are reconciled at different levels by multiresolution analysis. The distributional hypothesis meets the referential theory of meaning first at the semantic ordering of terms. While high-quality lexical resources enable such ordering in themselves, the ordering can benefit from data derived from a specific corpus being studied: composite semantic relatedness measures such as the Jiang-Conrath similarity operate this way. Once the semantic order is constructed, weights expressing statistical relationships between terms and documents are borrowed from the vector space model to form the basis for constructing hypothetical signals of content. By adjusting the length of the support of the basis functions, one may subtly balance the trade-off between distributional statistics and prior knowledge.

Based on a rigorous evaluation methodology and experimental results, the superiority of the proposed kernels in several application fields, including general domain data sets and text collections, is confirmed. However, a kernel based on a two-dimensional wavelet failed to improve performance; mainly due to issues with polysemy.

5.3 Future Work

Since the proposed kernels are able to incorporate prior knowledge in the classification process without taking any additional storage, they will be particularly useful in applications of emerging computing platforms, such as cloud computing and general purpose computing on graphics processors.

The concept of MapReduce is often associated with cloud computing (Dean and Ghemawat, 2004). When talking about cloud computing we refer to what is more precisely known as utility computing. Under this model, a user can dynamically provision any amount of computing resources from a (cloud) provider on demand and only pay for what is consumed (Lin and Dyer, 2010). Technically, this means that the user is paying for access to virtual machine instances that run a standard operating system. The virtualization technology enables the cloud provider to allocate available physical resources and enforce isolation between multiple users that may be sharing the same hardware. Once one or more virtual machine instances have been allocated, the user has full control over the resources and can use them for arbitrary computation. When the virtual instances are no longer needed, they are destroyed, thereby freeing up physical resources that can be redirected to other users. MapReduce provides the appropriate level of abstraction to this utility model by hiding the complexity of scaling to an arbitrary number of nodes which may fail. MapReduce draws on well-known principles in parallel and distributed computing and assembles them in a way to scale to collections of sizes unseen before. The proposed kernels would readily integrate with a MapReduce-based SVM implementation. Compared to the overhead of MapReduce, the additional computational need of these kernels is not that considerable. MapReduce jobs are typically data-bound, and the kernels ease the pressure on additional data transfer and storage, hence directly leading to cost savings.

General purpose progamming on graphical processing units promises extreme

speed-ups, in some cases the optimization enables up to seventy to eighty times faster execution. However, programming these processors is considerably different from utilizing multiple CPU cores. One fundamental difficulty is that the memory model is explicit, a developer has to juggle with several types of memories, and often faces very tight limits. For instance, on an nVidia G80-based graphical processing unit, the eight streaming cores of a simultaneous multiprocessor share sixteen kilobytes of high-speed memory. Since the proposed kernels do not need any additional storage, they can be a real benefit to systems that utilize graphics processing units.

Chapter 6

Appendix

6.1 Binary Classification Problems on General Data Sets

The following tables show the detailed experimental results on several data sets with baseline kernels and CSBF kernels. All problems are binary classification problems. For benchmarking methodology, refer to Section 3.7 and for interpreting the data, see Section 3.8.

Percentage of features	10 %	25 %	50 %	100 %
Linear	**0.853**	**0.853**	**0.824**	**0.824**
Polynomial 0.0 2	0.618	0.647	0.647	0.618
Polynomial 0.0 3	0.618	0.588	0.588	0.588
Polynomial 1.0 2	0.618	0.647	0.647	0.618
Polynomial 1.0 3	0.618	0.588	0.588	0.588
RBF 0.0	0.706	0.706	0.706	0.676
RBF 1.0	0.588	0.588	0.588	0.588
WSVM 1.0	0.588	0.588	0.588	0.588
WSVM 2.5	0.588	0.588	0.588	0.588
WSVM 5.0	0.588	0.588	0.588	0.588
WSVM 10.0	0.588	0.588	0.588	0.588

Table 6.1: Results with baseline kernels, Leukemia data set

Percentage of features	10 %	25 %	50 %	100 %
Linear	**0.602**	**0.598**	**0.600**	0.550
Polynomial 0.0 2	0.583	0.575	0.597	0.638
Polynomial 0.0 3	0.583	0.563	0.597	**0.645**
Polynomial 1.0 2	0.575	0.561	0.575	0.637
Polynomial 1.0 3	0.575	0.561	0.575	0.638
RBF 0.0	0.500	0.500	0.500	0.500
RBF 1.0	0.500	0.500	0.500	0.500
WSVM 1.0	0.500	0.500	0.500	0.500
WSVM 2.5	0.500	0.500	0.500	0.500
WSVM 5.0	0.500	0.500	0.500	0.500
WSVM 10.0	0.500	0.500	0.500	0.500

Table 6.2: Results with baseline kernels, Madelon data set

Percentage of features	10 %	25 %	50 %	100 %
Linear	0.951	0.975	0.976	0.976
Polynomial 0.0 2	**0.970**	**0.980**	**0.981**	**0.979**
Polynomial 0.0 3	0.966	0.973	0.975	0.976
Polynomial 1.0 2	**0.970**	**0.980**	**0.981**	**0.979**
Polynomial 1.0 3	0.966	0.973	0.975	0.976
RBF 0.0	0.500	0.500	0.500	0.500
RBF 1.0	0.500	0.500	0.500	0.500
WSVM 1.0	0.500	0.500	0.500	0.500
WSVM 2.5	0.500	0.500	0.500	0.500
WSVM 5.0	0.500	0.500	0.500	0.500
WSVM 10.0	0.500	0.500	0.500	0.500

Table 6.3: Results with baseline kernels, Gisette data set

Percentage of features	10 %	25 %	50 %	100 %
Correlation 0.5	**0.853**	0.853	0.824	0.824
Correlation 1.0	**0.853**	0.853	0.824	0.824
Correlation 2.0	**0.853**	0.853	0.824	0.824
Correlation 5.0	**0.853**	**0.912**	0.824	0.824
Correlation 10.0	0.824	**0.912**	**0.853**	**0.853**
Correlation 20.0	0.824	**0.912**	**0.853**	**0.853**
Euclidean 0.5	**0.853**	0.853	0.824	0.824
Euclidean 1.0	**0.853**	0.853	0.824	0.824
Euclidean 2.0	**0.853**	0.853	0.824	**0.853**
Euclidean 5.0	**0.853**	**0.912**	0.824	**0.853**
Euclidean 10.0	0.824	**0.912**	**0.853**	**0.853**
Euclidean 20.0	0.824	**0.912**	**0.853**	**0.853**
Mutual Inf 0.5	**0.853**	0.853	0.824	0.824
Mutual Inf 1.0	**0.853**	0.853	0.824	0.824
Mutual Inf 2.0	**0.853**	0.853	0.824	0.824
Mutual Inf 5.0	**0.853**	**0.912**	0.824	**0.853**
Mutual Inf 10.0	0.824	**0.912**	**0.853**	**0.853**
Mutual Inf 20.0	0.824	**0.912**	**0.853**	0.824

Table 6.4: Accuracy, Leukemia data set, equally spaced observations

Percentage of features	10 %	25 %	50 %	100 %
Correlation 0.5	**0.602**	**0.598**	**0.600**	**0.550**
Correlation 1.0	**0.602**	**0.598**	**0.600**	**0.550**
Correlation 2.0	0.563	0.486	0.500	0.516
Correlation 5.0	0.522	0.473	0.523	0.517
Correlation 10.0	0.520	0.548	0.540	0.500
Correlation 20.0	0.545	0.553	0.548	0.519
Euclidean 0.5	**0.602**	**0.598**	**0.600**	**0.550**
Euclidean 1.0	**0.602**	**0.598**	**0.600**	**0.550**
Euclidean 2.0	0.583	0.490	0.585	0.522
Euclidean 5.0	0.522	0.473	0.523	0.517
Euclidean 10.0	0.520	0.548	0.540	0.500
Euclidean 20.0	0.540	0.553	0.544	0.516
Mutual Inf 0.5	**0.602**	**0.598**	**0.600**	**0.550**
Mutual Inf 1.0	**0.602**	**0.598**	**0.600**	**0.550**
Mutual Inf 2.0	0.543	0.592	0.499	0.517
Mutual Inf 5.0	0.522	0.473	0.523	0.517
Mutual Inf 10.0	0.520	0.548	0.540	0.500
Mutual Inf 20.0	0.543	0.550	0.547	0.517

Table 6.5: Accuracy, Madelon data set, equally spaced observations

Percentage of features	10 %	25 %	50 %	100 %
Correlation 0.5	0.951	**0.975**	0.976	0.976
Correlation 1.0	0.951	**0.975**	0.976	0.976
Correlation 2.0	0.951	**0.975**	0.976	0.804
Correlation 5.0	0.950	0.965	0.967	0.967
Correlation 10.0	0.952	0.961	0.965	0.965
Correlation 20.0	0.940	0.930	0.942	**0.988**
Euclidean 0.5	**0.982**	0.956	**0.988**	0.976
Euclidean 1.0	0.940	0.956	0.974	0.976
Euclidean 2.0	0.951	**0.975**	0.976	0.814
Euclidean 5.0	0.950	0.965	0.967	0.967
Euclidean 10.0	0.920	0.887	0.909	0.965
Euclidean 20.0	0.963	0.971	0.981	0.984
Mutual Inf 0.5	0.953	0.904	0.987	0.976
Mutual Inf 1.0	0.889	0.904	0.948	0.976
Mutual Inf 2.0	0.951	**0.975**	0.976	0.812
Mutual Inf 5.0	0.950	0.965	0.967	0.967
Mutual Inf 10.0	0.952	0.961	0.965	0.965
Mutual Inf 20.0	0.940	0.901	0.928	0.985

Table 6.6: Accuracy, Gisette data set, equally spaced observations

Percentage of features	10 %	25 %	50 %	100 %
Correlation 0.5	0.853	**0.853**	0.824	0.794
Correlation 1.0	0.853	**0.853**	0.824	0.824
Correlation 2.0	**0.941**	0.824	0.912	0.824
Correlation 5.0	**0.941**	0.824	0.912	0.824
Correlation 10.0	**0.941**	0.824	**0.941**	0.824
Correlation 20.0	**0.941**	0.824	**0.941**	0.824
Euclidean 0.5	0.853	**0.853**	0.824	0.824
Euclidean 1.0	0.853	**0.853**	0.824	0.824
Euclidean 2.0	**0.941**	0.824	0.912	**0.853**
Euclidean 5.0	**0.941**	0.824	0.912	**0.853**
Euclidean 10.0	**0.941**	0.824	**0.941**	**0.853**
Euclidean 20.0	**0.941**	0.824	**0.941**	**0.853**
Mutual Inf 0.5	0.853	**0.853**	0.824	0.824
Mutual Inf 1.0	0.853	**0.853**	0.824	0.824
Mutual Inf 2.0	**0.941**	0.824	0.912	0.824
Mutual Inf 5.0	**0.941**	0.824	0.912	**0.853**
Mutual Inf 10.0	**0.941**	0.824	**0.941**	**0.853**
Mutual Inf 20.0	**0.941**	0.824	**0.941**	0.824

Table 6.7: Accuracy, Leukemia data set, randomly spaced observations

Percentage of features	10 %	25 %	50 %	100 %
Correlation 0.5	0.602	0.598	0.600	0.550
Correlation 1.0	0.602	0.598	0.600	0.550
Correlation 2.0	0.602	0.598	0.600	**0.612**
Correlation 5.0	0.502	0.542	0.443	0.505
Correlation 10.0	0.515	0.488	0.475	0.568
Correlation 20.0	0.432	0.446	0.400	0.461
Euclidean 0.5	0.600	0.598	0.598	0.549
Euclidean 1.0	0.602	0.598	0.600	0.550
Euclidean 2.0	0.502	0.542	0.443	0.505
Euclidean 5.0	0.449	0.458	0.409	0.504
Euclidean 10.0	0.515	0.488	0.475	0.568
Euclidean 20.0	0.432	0.446	0.400	0.461
Mutual Inf 0.5	**0.626**	**0.604**	**0.618**	0.568
Mutual Inf 1.0	0.602	0.598	0.600	0.550
Mutual Inf 2.0	0.502	0.542	0.443	0.505
Mutual Inf 5.0	0.431	0.500	0.360	0.492
Mutual Inf 10.0	0.515	0.488	0.475	0.568
Mutual Inf 20.0	0.432	0.446	0.400	0.461

Table 6.8: Accuracy, Madelon data set, randomly spaced observations

Percentage of features	10 %	25 %	50 %	100 %
Correlation 0.5	**0.951**	0.975	**0.976**	**0.976**
Correlation 1.0	**0.951**	0.975	**0.976**	**0.976**
Correlation 2.0	0.810	0.940	0.849	0.824
Correlation 5.0	0.928	0.945	0.945	0.945
Correlation 10.0	0.913	0.952	0.936	0.936
Correlation 20.0	0.921	**0.983**	0.947	0.942
Euclidean 0.5	**0.951**	0.975	**0.976**	**0.976**
Euclidean 1.0	**0.951**	0.975	**0.976**	**0.976**
Euclidean 2.0	0.765	0.822	0.874	0.861
Euclidean 5.0	0.928	0.945	0.945	0.945
Euclidean 10.0	0.913	0.952	0.936	0.936
Euclidean 20.0	0.918	0.977	0.939	0.934
Mutual Inf 0.5	**0.951**	0.975	**0.976**	**0.976**
Mutual Inf 1.0	**0.951**	0.975	**0.976**	**0.976**
Mutual Inf 2.0	0.833	0.805	0.830	0.910
Mutual Inf 5.0	0.928	0.945	0.945	0.945
Mutual Inf 10.0	0.913	0.952	0.936	0.936
Mutual Inf 20.0	0.920	0.979	0.942	0.933

Table 6.9: Accuracy, Gisette data set, randomly spaced observations

Percentage of features	10 %	25 %	50 %	100 %
Linear	**0.823**	**0.842**	**0.876**	**0.900**
Polynomial 0.0 2	0.775	0.786	0.829	0.874
Polynomial 0.0 3	0.732	0.729	0.777	0.818
Polynomial 1.0 2	0.780	0.795	0.841	0.886
Polynomial 1.0 3	0.745	0.750	0.796	0.835
RBF 0.0	0.412	0.470	0.522	0.554
RBF 1.0	0.266	0.159	0.090	0.027
WSVM 1.0	0.412	0.470	0.522	0.554
WSVM 2.5	0.266	0.159	0.090	0.027
WSVM 5.0	0.254	0.155	0.087	0.026
WSVM 10.0	0.262	0.158	0.088	0.026
LSI	0.705	0.769	0.781	0.750
Edge Counting	0.799	0.796	0.840	0.861

Table 6.10: Micro-Average F_1, Reuters Top-10, baseline kernels

6.2 Multiclass, Multilabel Classification Problems on Textual Data Sets

The following tables show the detailed experimental results on textual data sets with baseline kernels and CSBF kernels. All problems are multiclass, multilabel classification problems. For benchmarking methodology, refer to Section 4.6 and for interpreting the data, see Section 4.7.

Percentage of features	10 %	25 %	50 %	100 %
Linear	**0.735**	**0.764**	**0.804**	**0.843**
Polynomial 0.0 2	0.679	0.703	0.744	0.808
Polynomial 0.0 3	0.623	0.633	0.673	0.748
Polynomial 1.0 2	0.689	0.717	0.761	0.820
Polynomial 1.0 3	0.636	0.659	0.701	0.762
RBF 0.0	0.324	0.373	0.418	0.483
RBF 1.0	0.215	0.401	0.444	0.483
WSVM 1.0	0.324	0.373	0.418	0.483
WSVM 2.5	0.215	0.401	0.444	0.483
WSVM 5.0	0.205	0.390	0.428	0.474
WSVM 10.0	0.207	0.396	0.432	0.471
LSI	**0.735**	0.708	0.691	0.750
Edge Counting	**0.735**	0.733	0.758	0.788

Table 6.11: Micro-Average F_1, Reuters, baseline kernels

Percentage of features	10 %	25 %	50 %	100 %
Linear	0.514	0.557	**0.641**	0.672
Polynomial 0.0 2	0.503	0.562	0.578	0.599
Polynomial 0.0 3	**0.527**	**0.598**	0.600	0.463
Polynomial 1.0 2	0.503	0.562	0.578	0.617
Polynomial 1.0 3	0.527	0.589	0.606	0.617
RBF 0.0	0.018	0.024	0.025	0.026
RBF 1.0	0.020	0.025	0.025	0.002
WSVM 1.0	0.018	0.024	0.025	0.026
WSVM 2.5	0.020	0.025	0.025	**0.900**
WSVM 5.0	0.020	0.025	0.025	0.876
WSVM 10.0	0.020	0.025	0.025	0.876
LSI	0.448	0.456	0.615	0.604
Edge Counting	0.495	0.534	0.599	0.611

Table 6.12: Micro-Average F_1, 20News 50%, baseline kernels

Percentage of features	10 %	25 %	50 %	100 %
Linear	**0.518**	**0.653**	**0.671**	**0.693**
Polynomial 0.0 2	0.448	0.562	0.580	0.628
Polynomial 0.0 3	0.471	0.579	0.612	0.494
Polynomial 1.0 2	0.448	0.562	0.580	0.646
Polynomial 1.0 3	0.474	0.589	0.610	0.524
RBF 0.0	0.020	0.030	0.033	0.033
RBF 1.0	0.022	0.030	0.035	0.003
WSVM 1.0	0.020	0.030	0.033	0.033
WSVM 2.5	0.022	0.030	0.035	0.003
WSVM 5.0	0.022	0.030	0.034	0.003
WSVM 10.0	0.022	0.030	0.035	0.003
LSI	0.466	0.631	0.593	0.657
Edge Counting	0.497	0.623	0.652	0.651

Table 6.13: Micro-Average F_1, 20News 60%, baseline kernels

Percentage of features	10 %	25 %	50 %	100 %
Linear	0.516	0.595	**0.666**	**0.707**
Polynomial 0.0 2	0.550	0.585	0.605	0.650
Polynomial 0.0 3	0.582	**0.638**	0.641	0.513
Polynomial 1.0 2	0.550	0.585	0.605	0.671
Polynomial 1.0 3	**0.583**	0.608	0.634	0.545
RBF 0.0	0.035	0.042	0.041	0.042
RBF 1.0	0.038	0.044	0.043	0.003
WSVM 1.0	0.035	0.042	0.041	0.042
WSVM 2.5	0.038	0.044	0.043	0.003
WSVM 5.0	0.037	0.042	0.043	0.003
WSVM 10.0	0.037	0.043	0.042	0.003
LSI	0.440	0.542	0.636	0.644
Edge Counting	0.500	0.569	0.604	0.688

Table 6.14: Micro-Average F_1, 20News 70%, baseline kernels

Percentage of features	10 %	25 %	50 %	100 %
Correlation 0.5	0.823	0.842	0.876	0.900
Correlation 1.0	0.823	0.842	0.876	0.900
Correlation 2.0	0.863	0.887	0.943	0.900
Correlation 5.0	0.856	0.889	0.904	0.878
Correlation 10.0	0.753	0.791	0.815	0.882
Correlation 20.0	0.787	0.812	0.837	0.807
Edge Counting 0.5	0.823	0.842	0.876	0.900
Edge Counting 1.0	0.823	0.842	0.876	0.900
Edge Counting 2.0	0.854	0.897	0.885	0.900
Edge Counting 5.0	0.858	0.899	0.916	0.864
Edge Counting 10.0	0.743	0.761	0.755	0.825
Edge Counting 20.0	0.745	0.662	0.702	0.774
Jiang-Conrath 0.5	0.823	0.842	0.876	0.895
Jiang-Conrath 1.0	0.823	0.842	0.876	0.900
Jiang-Conrath 2.0	0.868	**1.000**	**0.984**	0.904
Jiang-Conrath 5.0	**0.955**	0.922	0.907	0.905
Jiang-Conrath 10.0	0.786	0.822	0.806	**0.906**
Jiang-Conrath 20.0	0.727	0.732	0.723	0.905

Table 6.15: Micro-Average F_1, Reuters Top-10, CSBF kernels

Percentage of features	10 %	25 %	50 %	100 %
Correlation 0.5	0.735	0.764	0.804	0.843
Correlation 1.0	0.735	0.764	0.804	0.843
Correlation 2.0	0.790	0.801	0.809	0.830
Correlation 5.0	0.740	0.779	0.848	0.825
Correlation 10.0	0.693	0.710	0.769	0.819
Correlation 20.0	0.735	0.764	0.804	0.843
Edge Counting 0.5	0.657	0.661	0.747	0.721
Edge Counting 1.0	0.735	0.764	0.804	0.843
Edge Counting 2.0	0.735	0.764	0.804	0.843
Edge Counting 5.0	0.778	0.832	0.843	**0.904**
Edge Counting 10.0	0.758	0.788	0.834	0.894
Edge Counting 20.0	0.677	0.699	0.754	0.785
Jiang-Conrath 0.5	0.701	0.723	0.724	0.659
Jiang-Conrath 1.0	0.735	0.764	0.804	0.843
Jiang-Conrath 2.0	**0.826**	0.829	**0.903**	0.833
Jiang-Conrath 5.0	0.801	**0.888**	0.863	0.823
Jiang-Conrath 10.0	0.694	0.734	0.774	0.811
Jiang-Conrath 20.0	0.735	0.764	0.804	0.843

Table 6.16: Micro-Average F_1, Reuters, CSBF kernels

Percentage of features	10 %	25 %	50 %	100 %
Correlation 0.5	0.514	0.557	0.641	0.672
Correlation 1.0	0.514	0.557	0.641	0.672
Correlation 2.0	0.514	0.557	0.641	0.672
Correlation 5.0	0.492	0.526	0.584	0.633
Correlation 10.0	0.446	0.475	0.567	0.610
Correlation 20.0	0.373	0.428	0.498	0.591
Edge Counting 0.5	0.514	0.557	0.641	0.672
Edge Counting 1.0	0.514	0.557	0.641	0.672
Edge Counting 2.0	0.547	0.585	0.673	0.696
Edge Counting 5.0	0.537	0.580	0.672	**0.706**
Edge Counting 10.0	0.454	0.492	0.576	0.574
Edge Counting 20.0	0.409	0.417	0.513	0.528
Jiang-Conrath 0.5	0.514	0.557	0.641	0.672
Jiang-Conrath 1.0	0.514	0.557	0.641	0.672
Jiang-Conrath 2.0	0.550	0.584	0.670	0.663
Jiang-Conrath 5.0	**0.580**	**0.602**	**0.711**	0.649
Jiang-Conrath 10.0	0.504	0.508	0.625	0.634
Jiang-Conrath 20.0	0.466	0.487	0.586	0.598

Table 6.17: Micro-Average F_1, 20News 50%, CSBF kernels

Percentage of features	10 %	25 %	50 %	100 %
Correlation 0.5	0.518	0.653	0.671	0.693
Correlation 1.0	0.518	0.653	0.671	0.693
Correlation 2.0	0.518	0.653	0.671	0.693
Correlation 5.0	0.488	0.631	0.618	0.664
Correlation 10.0	0.487	0.604	0.578	0.648
Correlation 20.0	0.456	0.548	0.603	0.624
Edge Counting 0.5	0.518	0.653	0.671	0.693
Edge Counting 1.0	0.518	0.653	0.671	0.693
Edge Counting 2.0	0.541	**0.709**	0.708	**0.735**
Edge Counting 5.0	0.558	0.674	0.711	0.707
Edge Counting 10.0	0.455	0.568	0.587	0.595
Edge Counting 20.0	0.451	0.648	0.586	0.603
Jiang-Conrath 0.5	0.518	0.653	0.671	0.693
Jiang-Conrath 1.0	0.518	0.653	0.671	0.693
Jiang-Conrath 2.0	0.537	0.668	0.713	0.680
Jiang-Conrath 5.0	**0.581**	0.706	**0.718**	0.665
Jiang-Conrath 10.0	0.501	0.618	0.627	0.649
Jiang-Conrath 20.0	0.508	0.607	0.564	0.611

Table 6.18: Micro-Average F_1, 20News 60%, CSBF kernels

Percentage of features	10 %	25 %	50 %	100 %
Correlation 0.5	0.516	0.595	0.666	0.707
Correlation 1.0	0.516	0.595	0.666	0.707
Correlation 2.0	0.516	0.595	0.666	0.707
Correlation 5.0	0.491	0.563	0.636	0.679
Correlation 10.0	0.499	0.536	0.573	0.632
Correlation 20.0	0.454	0.552	0.544	0.578
Edge Counting 0.5	0.516	0.595	0.666	0.707
Edge Counting 1.0	0.516	0.595	0.666	0.707
Edge Counting 2.0	0.535	0.636	0.687	**0.743**
Edge Counting 5.0	0.544	0.606	0.683	0.734
Edge Counting 10.0	0.468	0.530	0.582	0.655
Edge Counting 20.0	0.424	0.582	0.525	0.571
Jiang-Conrath 0.5	0.516	0.595	0.666	0.707
Jiang-Conrath 1.0	0.516	0.595	0.666	0.707
Jiang-Conrath 2.0	0.524	0.626	0.681	0.691
Jiang-Conrath 5.0	**0.602**	**0.700**	**0.762**	0.676
Jiang-Conrath 10.0	0.486	0.567	0.630	0.660
Jiang-Conrath 20.0	0.460	0.561	0.588	0.635

Table 6.19: Micro-Average F_1, 20News 70%, CSBF kernels

Percentage of features	10 %	25 %	50 %	100 %
Linear	**0.441**	**0.483**	**0.525**	**0.627**
Polynomial 0.0 2	0.406	0.445	0.481	0.559
Polynomial 0.0 3	0.356	0.386	0.418	0.471
Polynomial 1.0 2	0.416	0.452	0.489	0.569
Polynomial 1.0 3	0.363	0.406	0.438	0.481
RBF 0.0	0.021	0.025	0.032	0.051
RBF 1.0	0.042	0.027	0.033	0.051
WSVM 1.0	0.021	0.025	0.032	0.051
WSVM 2.5	0.042	0.027	0.033	0.051
WSVM 5.0	0.042	0.027	0.032	0.050
WSVM 10.0	0.042	0.026	0.032	0.049
LSI	**0.441**	0.441	0.446	0.539
EdgeCountingLin	**0.441**	0.472	0.483	0.602

Table 6.20: Macro-Average F_1, Reuters, baseline kernels

Percentage of features	10 %	25 %	50 %	100 %
Linear	**0.699**	**0.722**	**0.803**	**0.825**
Polynomial 0.0 2	0.668	0.662	0.733	0.767
Polynomial 0.0 3	0.607	0.590	0.656	0.673
Polynomial 1.0 2	0.673	0.680	0.755	0.790
Polynomial 1.0 3	0.626	0.616	0.694	0.691
RBF 0.0	0.174	0.204	0.234	0.000
RBF 1.0	0.151	0.109	0.085	0.000
WSVM 1.0	0.174	0.204	0.234	0.000
WSVM 2.5	0.151	0.109	0.085	0.000
WSVM 5.0	0.145	0.106	0.082	0.000
WSVM 10.0	0.149	0.109	0.083	0.000
LSI	0.622	0.636	0.708	0.748
EdgeCountingLin	0.685	0.681	0.787	0.775

Table 6.21: Macro-Average F_1, Reuters Top-10, baseline kernels

Percentage of features	10 %	25 %	50 %	100 %
Linear	**0.548**	**0.589**	**0.633**	**0.669**
Polynomial 0.0 2	0.404	0.522	0.562	0.594
Polynomial 0.0 3	0.413	0.561	0.600	0.463
Polynomial 1.0 2	0.404	0.522	0.562	0.612
Polynomial 1.0 3	0.418	0.538	0.609	0.612
RBF 0.0	0.000	0.000	0.000	0.000
RBF 1.0	0.000	0.000	0.000	0.000
WSVM 1.0	0.000	0.000	0.000	0.000
WSVM 2.5	0.000	0.000	0.000	0.000
WSVM 5.0	0.000	0.000	0.000	0.000
WSVM 10.0	0.000	0.000	0.000	0.000
LSI	0.486	0.543	0.590	0.595
EdgeCountingLin	0.517	0.559	0.583	0.644

Table 6.22: Macro-Average F_1, 20News 50%, baseline kernels

Percentage of features	10 %	25 %	50 %	100 %
Linear	**0.615**	**0.651**	**0.672**	**0.692**
Polynomial 0.0 2	0.466	0.510	0.605	0.624
Polynomial 0.0 3	0.484	0.521	0.634	0.494
Polynomial 1.0 2	0.466	0.510	0.605	0.642
Polynomial 1.0 3	0.490	0.552	0.649	0.523
RBF 0.0	0.000	0.000	0.000	0.000
RBF 1.0	0.000	0.000	0.000	0.000
WSVM 1.0	0.000	0.000	0.000	0.000
WSVM 2.5	0.000	0.000	0.000	0.000
WSVM 5.0	0.000	0.000	0.000	0.000
WSVM 10.0	0.000	0.000	0.000	0.000
LSI	0.590	0.578	0.625	0.633
EdgeCountingLin	0.590	0.609	0.641	0.672

Table 6.23: Macro-Average F_1, 20News 60%, baseline kernels

Percentage of features	10 %	25 %	50 %	100 %
Linear	**0.545**	**0.636**	**0.669**	**0.707**
Polynomial 0.0 2	0.457	0.499	0.619	0.646
Polynomial 0.0 3	0.475	0.537	0.655	0.513
Polynomial 1.0 2	0.457	0.499	0.619	0.667
Polynomial 1.0 3	0.497	0.529	0.639	0.543
RBF 0.0	0.037	0.032	0.040	0.042
RBF 1.0	0.039	0.035	0.044	0.003
WSVM 1.0	0.037	0.032	0.040	0.042
WSVM 2.5	0.039	0.035	0.044	0.003
WSVM 5.0	0.039	0.034	0.042	0.003
WSVM 10.0	0.038	0.035	0.043	0.003
LSI	0.509	0.580	0.614	0.618
EdgeCountingLin	0.526	0.598	0.614	0.653

Table 6.24: Macro-Average F_1, 20News 70%, baseline kernels

Percentage of features	10 %	25 %	50 %	100 %
Correlation 0.5	0.699	0.722	0.803	0.826
Correlation 1.0	0.699	0.722	0.803	0.826
Correlation 2.0	0.708	0.771	0.818	0.826
Correlation 5.0	0.745	0.757	**0.861**	0.791
Correlation 10.0	0.651	0.662	0.753	0.790
Correlation 20.0	0.654	0.679	0.745	0.722
EdgeCountingCSBF 0.5	0.699	0.722	0.803	0.826
EdgeCountingCSBF 1.0	0.699	0.722	0.803	0.826
EdgeCountingCSBF 2.0	0.706	0.759	0.853	0.826
EdgeCountingCSBF 5.0	0.737	0.753	0.823	0.773
EdgeCountingCSBF 10.0	0.667	0.696	0.667	0.784
EdgeCountingCSBF 20.0	0.507	0.628	0.679	0.785
Jiang-Conrath 0.5	0.699	0.722	0.803	0.817
Jiang-Conrath 1.0	0.699	0.722	0.803	0.825
Jiang-Conrath 2.0	0.775	**0.849**	0.845	**0.833**
Jiang-Conrath 5.0	**0.781**	0.781	0.853	0.833
Jiang-Conrath 10.0	0.690	0.699	0.783	0.830
Jiang-Conrath 20.0	0.638	0.681	0.679	0.825

Table 6.25: Macro-Average F_1, Reuters Top-10, CSBF kernels

Percentage of features	10 %	25 %	50 %	100 %
Correlation 0.5	0.441	0.483	0.525	0.627
Correlation 1.0	0.441	0.483	0.525	0.627
Correlation 2.0	**0.473**	0.524	0.564	0.000
Correlation 5.0	0.457	0.509	0.560	0.000
Correlation 10.0	0.423	0.461	0.518	0.000
Correlation 20.0	0.441	0.483	0.525	0.627
EdgeCountingCSBF 0.5	0.407	0.466	0.467	0.536
EdgeCountingCSBF 1.0	0.441	0.483	0.525	0.627
EdgeCountingCSBF 2.0	0.441	0.483	0.525	0.627
EdgeCountingCSBF 5.0	0.460	0.522	0.542	0.652
EdgeCountingCSBF 10.0	0.462	0.506	0.539	**0.663**
EdgeCountingCSBF 20.0	0.374	0.414	0.471	0.576
Jiang-Conrath 0.5	0.430	0.356	0.430	0.469
Jiang-Conrath 1.0	0.441	0.483	0.525	0.627
Jiang-Conrath 2.0	0.472	0.520	0.558	0.000
Jiang-Conrath 5.0	0.470	**0.570**	**0.617**	0.000
Jiang-Conrath 10.0	0.413	0.441	0.518	0.000
Jiang-Conrath 20.0	0.441	0.483	0.525	0.627

Table 6.26: Macro-Average F_1, Reuters, CSBF kernels

Percentage of features	10 %	25 %	50 %	100 %
Correlation 0.5	0.548	0.589	0.633	0.669
Correlation 1.0	0.548	0.589	0.633	0.669
Correlation 2.0	0.548	0.589	0.633	0.669
Correlation 5.0	0.520	0.554	0.627	0.636
Correlation 10.0	0.498	0.530	0.550	0.604
Correlation 20.0	0.398	0.456	0.544	0.520
EdgeCountingCSBF 0.5	0.548	0.589	0.633	0.669
EdgeCountingCSBF 1.0	0.548	0.589	0.633	0.669
EdgeCountingCSBF 2.0	0.571	0.624	0.667	**0.716**
EdgeCountingCSBF 5.0	0.577	0.627	**0.680**	0.686
EdgeCountingCSBF 10.0	0.482	0.494	0.576	0.609
EdgeCountingCSBF 20.0	0.424	0.511	0.518	0.498
Jiang-Conrath 0.5	0.548	0.589	0.633	0.669
Jiang-Conrath 1.0	0.548	0.589	0.633	0.669
Jiang-Conrath 2.0	**0.596**	0.635	0.655	0.661
Jiang-Conrath 5.0	0.588	**0.666**	0.673	0.648
Jiang-Conrath 10.0	0.512	0.550	0.598	0.634
Jiang-Conrath 20.0	0.517	0.566	0.550	0.645

Table 6.27: Macro-Average F_1, 20News 50%, CSBF kernels

Percentage of features	10 %	25 %	50 %	100 %
Correlation 0.5	0.615	0.651	0.672	0.692
Correlation 1.0	0.615	0.651	0.672	0.692
Correlation 2.0	0.615	0.651	0.672	0.692
Correlation 5.0	0.579	0.628	0.621	0.675
Correlation 10.0	0.499	0.571	0.595	0.610
Correlation 20.0	0.443	0.519	0.584	0.577
EdgeCountingCSBF 0.5	0.615	0.651	0.672	0.692
EdgeCountingCSBF 1.0	0.615	0.651	0.672	0.692
EdgeCountingCSBF 2.0	0.654	0.709	0.695	0.723
EdgeCountingCSBF 5.0	0.643	0.675	0.714	**0.723**
EdgeCountingCSBF 10.0	0.575	0.565	0.625	0.632
EdgeCountingCSBF 20.0	0.506	0.553	0.603	0.642
Jiang-Conrath 0.5	0.615	0.651	0.672	0.692
Jiang-Conrath 1.0	0.615	0.651	0.672	0.692
Jiang-Conrath 2.0	0.635	0.706	**0.714**	0.681
Jiang-Conrath 5.0	**0.694**	**0.710**	0.684	0.666
Jiang-Conrath 10.0	0.596	0.613	0.639	0.650
Jiang-Conrath 20.0	0.555	0.590	0.591	0.685

Table 6.28: Macro-Average F_1, 20News 60%, CSBF kernels

Percentage of features	10 %	25 %	50 %	100 %
Correlation 0.5	0.545	0.636	0.669	0.707
Correlation 1.0	0.545	0.636	0.669	0.707
Correlation 2.0	0.545	0.636	0.669	0.707
Correlation 5.0	0.563	0.675	0.696	0.756
Correlation 10.0	0.573	0.655	0.712	0.651
Correlation 20.0	0.580	0.648	0.694	0.569
EdgeCountingCSBF 0.5	0.545	0.636	0.669	0.707
EdgeCountingCSBF 1.0	0.545	0.636	0.669	0.707
EdgeCountingCSBF 2.0	0.587	0.644	0.706	0.738
EdgeCountingCSBF 5.0	0.500	0.609	0.643	0.721
EdgeCountingCSBF 10.0	0.561	0.643	0.679	0.660
EdgeCountingCSBF 20.0	0.460	0.519	0.577	0.700
Jiang-Conrath 0.5	0.545	0.636	0.669	0.707
Jiang-Conrath 1.0	0.545	0.636	0.669	0.707
Jiang-Conrath 2.0	0.587	0.653	0.696	0.693
Jiang-Conrath 5.0	0.590	0.644	0.739	0.677
Jiang-Conrath 10.0	0.493	0.589	0.652	0.662
Jiang-Conrath 20.0	**0.609**	**0.695**	**0.756**	**0.793**

Table 6.29: Macro-Average F_1, 20News 70%, CSBF kernels

References

Abdou, S. and J. Savoy. 2008. Searching in Medline: Query expansion and manual indexing evaluation. *Information Processing & Management*, 44(2):781–789.

Abramovich, F., T.C. Bailey, and T. Sapatinas. 2000. Wavelet analysis and its statistical applications. *The Statistician*, 49(1):1–29.

Aha, D.W. 1992. Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies*, 36(2):267–287.

Aha, D.W. and R.L. Bankert. 1994. Feature selection for case-based classification of cloud types: An empirical comparison. In *Working Notes of the AAAI-94 Workshop on Case-Based Reasoning*, pages 106–112, Seattle, WA, USA, July. AAAI Press, Menlo Park, CA, USA.

Aha, D.W., D. Kibler, and M.K. Albert. 1991. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66.

Alexandrov, T., J. Decker, B. Mertens, A.M. Deelder, R.A.E.M. Tollenaar, P. Maass, and H. Thiele. 2009. Biomarker discovery in MALDI-TOF serum protein profiles using discrete wavelet transformation. *Bioinformatics*, 25(5):643.

Allen, D.M. 1974. The relationship between variable selection and data augmentation and a method for prediction. *Technometrics*, 16(1):125–127.

Allen, J.B. and L.R. Rabiner. 1977. A unified approach to short-time Fourier analysis and synthesis. *Proceedings of the IEEE*, 65(11):1558–1564.

Almuallim, H. and T.G. Dietterich. 1991. Learning with many irrelevant features. In *Proceedings of AAAI-91, 9th National Conference on Artificial Intelligence*, volume 2, pages 547–552, Anaheim, CA, USA. Morgan Kaufmann Publishers, San Francisco, CA, USA.

Almuallim, H. and T.G. Dietterich. 1992. Efficient algorithms for identifying relevant features. In *Proceedings of AI-92, 9th Canadian Conference on Artificial Intelligence*, pages 38–45, Vancouver, BC, Canada, May. Morgan Kaufmann Publishers, San Francisco, CA, USA.

Ankerst, M., M.M. Breunig, H.P. Kriegel, and J. Sander. 1999. OPTICS: Ordering points to identify the clustering structure. In *Proceedings of SIGMOD-99, International Conference on Management of Data*, pages 49–60. ACM Press, New York, NY, USA.

Apté, C., F. Damerau, and S.M. Weiss. 1994a. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12(3):233–251.

Apté, C., F. Damerau, and S.M. Weiss. 1994b. Towards language independent automated learning of text categorization models. In *Proceedings of SIGIR-94, 17th International Conference on Research and Development in Information Retrieval*, Dublin, Ireland, July. ACM Press, New York, NY, USA.

Baeza-Yates, R. and B. Ribeiro-Neto. 1999. *Modern Information Retrieval*. Addison-Wesley, Harlow, UK.

Baker, L.D. and A.K. McCallum. 1998. Distributional clustering of words for text classification. In *Proceedings of SIGIR-98, 21st International Conference on Research and Development in Information Retrieval*, pages 96–103, Melbourne, Australia, August. ACM Press, New York, NY, USA.

Banerjee, S. and T. Pedersen. 2002. An adapted Lesk algorithm for word sense disambiguation using WordNet. In *Proceedings of CICLing-02, 3rd International Conference on Intelligent Text Processing and Computational Linguistics*, Lecture Notes in Computer Science, pages 136–145, Mexico City, Mexico, February. Springer.

Basili, R., M. Cammisa, and A. Moschitti. 2005. Effective use of WordNet seman-

tics via kernel-based learning. In *Proceedings of CoNLL-05, 9th Conference on Computational Natural Language Learning*, pages 1–8, Ann Arbor, MI, USA, June. ACL, Morristown, NJ, USA.

Beckmann, N., H.P. Kriegel, R. Schneider, and B. Seeger. 1990. The R*-tree: an efficient and robust access method for points and rectangles. *SIGMOD Record*, 19(2):322–331.

Bekkerman, R., R. El-Yaniv, N. Tishby, Y. Winter, I. Guyon, and A. Elisseeff. 2003. Distributional word clusters vs. words for text categorization. *Journal of Machine Learning Research*, 3(7-8):1183–1208.

Berchtold, S., D.A. Keim, and H.P. Kriegel. 2001. The X-tree: An index structure for high-dimensional data. In K. Jeffay and H.J. Zhang, editors, *Readings in multimedia computing and networking*. Morgan Kaufmann, page 451.

Berger, A.L., V.J. Della Pietra, and S.A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Berrey, L.V. and G. Carruth. 1962. *Roget's International Thesaurus*. Thomas Y. Crowell, New York, NY, USA.

Berry, M.W., Z. Drmac, and E.R. Jessup. 1999. Matrices, vector spaces, and information retrieval. *SIAM Review*, 41(2):335–362.

Berry, M.W., S.T. Dumais, and G.W. O'Brien. 1995. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573–595.

Bi, J., K.P. Bennett, M. Embrechts, C.M. Breneman, M. Song, I. Guyon, and A. Elisseeff. 2003. Dimensionality reduction via sparse support vector machines. *Journal of Machine Learning Research*, 3(7-8):1229–1243.

Bloomfield, L. 1933. *Language*. Holt, Reinhart and Winston, New York, NY, USA.

Breiman, L., J.H. Friedman, R.A. Olshen, and C.J. Stone. 1984. *Classification and Regression Trees*. Wadsworth & Brooks, Monterey, CA, USA.

Buckley, C., G. Salton, and J. Allan. 1994. The effect of adding relevance information in a relevance feedback environment. In *Proceedings of SIGIR-94, 17th International Conference on Research and Development in Information Retrieval*, Dublin, Ireland, July. ACM Press, New York, NY, USA.

Budanitsky, A. and G. Hirst. 2001. Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures. In *Workshop on WordNet and Other Lexical Resources, Second meeting of the North American section of ACL*, pages 29–34. ACL, Morristown, NJ, USA.

Budanitsky, A. and G. Hirst. 2006. Evaluating WordNet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47.

Burges, C.J.C. 1998. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167.

Cao, B., D. Shen, J.T. Sun, Q. Yang, and Z. Chen. 2007. Feature selection in a kernel space. In *Proceedings of ICML-07, 24th International Conference on Machine Learning*, pages 121–128, Corvallis, OR, USA, June. ACM Press, New York, NY, USA.

Cardie, C. 1993. Using decision trees to improve case-based learning. In *Proceedings of ICML-93, 10th International Conference on Machine Learning*, pages 25–32, Amherst, MA, USA. Morgan Kaufmann Publishers, San Francisco, CA, USA.

Carnap, R. 1947. *Meaning and Necessity: A Study in Semantics and Modal Logic*. University Of Chicago Press, Chicago, IL, USA.

Caropreso, M.F., S. Matwin, and F. Sebastiani. 2001. A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization. In A.G. Chin, editor, *Text Databases and Document Management: Theory and Practice*. Idea Group Publishing, Hershey, PA, USA, pages 78–102.

Caruana, R., V.R. de Sa, I. Guyon, and A. Elisseeff. 2003. Benefitting from the variables that variable selection discards. *Journal of Machine Learning Research*, 3(7-8):1245–1264.

Caruana, R. and D. Freitag. 1994. Greedy attribute selection. In *Proceedings of ICML-94, 11th International Conference on Machine Learning*, pages 28–36, New Brunswick, NJ, USA, July. Morgan Kaufmann Publishers, San Francisco, CA, USA.

Chang, Chih-Chung and Chih-Jen Lin, 2001. *LIBSVM: A library for support vector machines*. `http://www.csie.ntu.edu.tw/\~{}cjlin/libsvm`.

Chen, G.Y. and P. Bhattacharya. 2006. Function dot product kernels for support vector machine. In *Proceedings of ICPR-06, 18th International Conference on Pattern Recognition*, volume 2.

Chen, G.Y. and W.F. Xie. 2007. Pattern recognition with SVM and dual-tree complex wavelets. *Image and Vision Computing*, 25(6):960–966.

Chui, C.K. and J. Lian. 1996. A study of orthonormal multi-wavelets. *Applied Numerical Mathematics*, 20(3):273–298.

Church, K.W. and P. Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

Ciaccia, P., M. Patella, and P. Zezula. 1997. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of VLDB-97, 23rd International Conference on Very Large Data Bases*, pages 426–435, Athens, Greece, August. IEEE.

Cilibrasi, R. and P.M.B. Vitanyi. 2007. The Google similarity distance. *IEEE Transactions on Knowledge and Data Engineering*, 19(3).

Cohen, W.W. and Y. Singer. 1996. Context-sensitive learning methods for text categorization. In *Proceedings of SIGIR-96, 19th International Conference on*

Research and Development in Information Retrieval, pages 307–315, Zürich, Switzerland, August. ACM Press, New York, NY, USA.

Cristianini, N. and J. Shawe-Taylor. 2000. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, New York, NY, USA.

Cristianini, N., J. Shawe-Taylor, and H. Lodhi. 2002. Latent semantic kernels. *Journal of Intelligent Information Systems*, 18(2):127–152.

Csiszár, I. 1996. Maxent, mathematics, and information theory. In K. Hanson and R. Silver, editors, *Maximum Entropy and Bayesian Methods*. Kluwer Academic Publishers Norwell, MA, USA.

Dagan, I., Y. Karov, and D. Roth. 1997. Mistake-driven learning in text categorization. In *Proceedings of EMNLP-97, 2nd Conference on Empirical Methods in Natural Language Processing*, pages 55–63, Providence, RI, US.

Daubechies, I. 1992. *Ten lectures on wavelets*. Society for Industrial Mathematics, Philadelphia, PA, USA.

Daubechies, I., B. Han, A. Ron, and Z. Shen. 2003. Framelets: MRA-based constructions of wavelet frames. *Applied and Computational Harmonic Analysis*, 14(1):1–46.

Dave, K., S. Lawrence, and D.M. Pennock. 2003. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of IW3C-03, 12th International World Wide Web Conference*, pages 519–528, Budapest, Hungary, May. ACM Press, New York, NY, USA.

Davidson, G.S., B.N. Wylie, and K.W. Boyack. 2001. Cluster stability and the use of noise in interpretation of clustering. In *Proceedings of Vis-01, IEEE Information Visualization*, pages 23–30, San Diego, CA, USA, October.

Dawson, A. and A. Slevin. 2008. Repository case history: University of Strathclyde Strathprints.

Dean, J. and S. Ghemawat. 2004. MapReduce: Simplified data processing on large clusters. In *Proceedings of OSDI-04, 6th International Symposium on Operating Systems Design & Implementation*, San Francisco, CA, USA, December. ACM Press, New York, NY, USA.

Deerwester, S., S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Della Pietra, S., V. Della Pietra, and J. Lafferty. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393.

Devijver, P.A. and J. Kittler. 1982. *Pattern Recognition: A Statistical Approach*. Prentice-Hall, Upper Saddle River, NJ, USA.

Dietterich, T.G. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923.

Domingos, F. 1997. Control-sensitive feature selection for lazy learners. *Artificial Intelligence Review*, 11(1):227–253.

Dominich, S. and T. Kiezer. 2007. A measure theoretic approach to information retrieval. *Journal of the American Society for Information Science and Technology*, 58(8):1108–1122.

Donoho, S. and L. Rendell. 1996. Constructive induction using fragmentary knowledge. In *Proceedings of ICML-96, 13th International Conference on Machine Learning*, pages 113–121, Bari, Italy, July. Morgan Kaufmann Publishers, San Francisco, CA, USA.

Duda, R.O., P.E. Hart, and D.G. Stork. 2000. *Pattern Classification*. John Wiley & Sons, New York, NY, USA.

Dumais, S., J. Platt, D. Heckerman, and M. Sahami. 1998. Inductive learning algorithms and representations for text categorization. In *Proceedings*

of CIKM-98, 7th International Conference on Information and Knowledge Management, pages 148–155, Bethesda, MD, USA. ACM Press, New York, NY, USA.

Ester, M., H.P. Kriegel, J. Sander, and X. Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of SIGKDD-96, 2nd International Conference on Knowledge Discovery and Data Mining*, volume 96, pages 226–231, Portland, OR, USA, August. ACM Press, New York, NY, USA.

Everitt, B. 1992. *The Analysis of Contingency Tables*. Chapman and Hall, New York, NY, USA.

Fellbaum, C. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, USA.

Fonseca, E.S., R.C. Guido, P.R. Scalassara, C.D. Maciel, and J.C. Pereira. 2007. Wavelet time-frequency analysis and least squares support vector machines for the identification of voice disorders. *Computers in Biology and Medicine*, 37(4):571–578.

Fonseca, E.S., R.C. Guido, A.C. Silvestre, and J.C. Pereira. 2005. Discrete wavelet transform and support vector machine applied to pathological voice signals identification. In *Proceedings of ISM-05, 7th IEEE International Symposium on Multimedia*, pages 785–789. IEEE Computer Society, December.

Forman, G., I. Guyon, and A. Elisseeff. 2003. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3(7-8):1289–1305.

Frege, G. 1948. Sense and reference. *The philosophical review*, 57(3):209–230.

Furey, T.S., N. Cristianini, N. Duffy, D.W. Bednarski, M. Schummer, and D. Haussler. 2000. Support vector machine classification and validation of cancer tis-

sue samples using microarray expression data. *Bioinformatics*, 16(10):906–914.

Gabrilovich, E. and S. Markovitch. 2005. Feature generation for text categorization using world knowledge. In *Proceedings of IJCAI-05, 19th International Joint Conference on Artificial Intelligence*, volume 19, Edinburgh, UK. Lawrence Erlbaum Associates.

Gallant, S. I. 1991. A practical approach for representing context and for performing word sense disambiguation using neural networks. *Neural Computation*, 3:293–309.

Gallant, S. I., W. R. Caid, J. Carleton, R. Hecht-Nielsen, K. P. Qing, and D. Sudbeck. 1992. HNC's MatchPlus system. In *Proceedings of TREC-92, 1st Text Retrieval Conference*, volume 2, pages 34–38, Gaithersburg, MD, USA, November. National Institute of Standards and Technology.

Gamberger, D. and N. Lavrac. 1997. Conditions for Occam's razor applicability and noise elimination. In W. Gerstner, A. Germond, M. Hasler, and J.D. Nicoud, editors, *Proccedings of ECML-97, 9th European Conference on Machine Learning*, pages 108–123, Prague, Czech Republic, April. Springer.

Globerson, A., N. Tishby, I. Guyon, and A. Elisseeff. 2003. Sufficient dimensionality reduction. *Journal of Machine Learning Research*, 3(7-8):1307–1331.

Gohberg, I. and S. Goldberg. 1980. *Basic Operator Theory*. Birkhäuser, Boston, MA, USA.

Golub, G. and C. Reinsch. 1971. *Handbook for Automatic Computation II, Linear Algebra*. Springer-Verlag, New York, NY, USA.

Gomez-Perez, A. 1995. Some ideas and examples to evaluate ontologies. In *Proceedings of CAIA-95, 11th Conference on Artificial Intelligence for Applications*, pages 299–305, Los Angeles, CA, USA, February.

Gonzalo, J., F. Verdejo, I. Chugur, and J. Cigarran. 1998. Indexing with WordNet synsets can improve text retrieval. In *Proceedings of COLING-ACL-98, 17th International Conference on Computational Linguistics*, volume 98, pages 38–44, Montréal, Québec, Canada, August. ACL, Morristown, NJ, USA.

Grefenstette, G. 1992. Use of syntactic context to produce term association lists for text retrieval. In *Proceedings of SIGIR-92, 15th International Conference on Research and Development in Information Retrieval*, pages 89–97, Copenhagen, Denmark, June. ACM Press, New York, NY, USA.

Gruber, T.R. 1995. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43(5/6):907–928.

Guyon, I., A. Elisseefi, and L.P. Kaelbling. 2003. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3(7-8):1157–1182.

Guyon, I., S. Gunn, A. Ben-Hur, and G. Dror. 2005. Result analysis of the NIPS 2003 feature selection challenge. *Advances in Neural Information Processing Systems*, 17:545–552.

Hall, M., E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. 2009. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1).

Harnad, S.R. 1987. *Categorical Perception: The Groundwork of Cognition*. Cambridge University Press, New York, NY, USA.

Harris, Z. 1970. Distributional structure. In Z. Harris, editor, *Papers in structural and transformational Linguistics*, Formal Linguistics. Humanities Press, New York, NY, USA, pages 775–794.

Hastie, T., R. Tibshirani, and J. Friedman. 2001. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

Hayes, P.J. and S.P. Weinstein. 1990. CONSTRUE/TIS: A system for content-based indexing of a database of news stories. In *Proceedings of IAAI-90, 2nd Conference on Innovative Applications of Artificial Intelligence*, pages 49–64, Washington, D.C., USA, May. AAAI Press, Menlo Park, CA, USA.

Hirst, G. and D. St-Onge. 1998. Lexical chains as representations of context for the detection and correction of malapropisms. In C. Fellbaum, editor, *Wordnet: An Electronic Lexical Database*. MIT Press, New York, NY, USA.

Hoenkamp, E. 2003. Unitary operators on the document space. *Journal of the American Society for Information Science and Technology*, 54(4):314–320.

Hosseini, P.T., F. Almasganj, T. Emami, R. Behroozmand, S. Gharibzade, and F. Torabinezhad. 2008. Local discriminant wavelet packet basis for voice pathology classification. In *Proceedings of ICBBE-08, 2nd International Conference on Bioinformatics and Biomedical Engineering*, pages 2052–2055, Shanghai, China, May.

Hotho, A., S. Staab, and G. Stumme. 2003. WordNet improves text document clustering. In *Proceedings of SIGIR-03, 26th International Conference on Research and Development in Information Retrieval*, Toronto, Canada, July. ACM Press, New York, NY, USA.

Hsu, C.W. and C.J. Lin. 2002a. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425.

Hsu, C.W. and C.J. Lin. 2002b. A simple decomposition method for support vector machines. *Machine Learning*, 46(1):291–314.

Hull, D. 1994. Improving text retrieval for the routing problem using latent semantic indexing. In *Proceedings of SIGIR-94, 17th International Conference on Research and Development in Information Retrieval*, Dublin, Ireland, July. ACM Press, New York, NY, USA.

Jain, A.K., M.N. Murty, and P.J. Flynn. 1999. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323.

Jarmasz, M. and S. Szpakowicz. 2003. Roget's thesaurus and semantic similarity. In *Proceedings of RANLP-03, 4th International Conference on Recent Advances in Natural Language Processing*, Borovets, Bulgaria, September. John Benjamins Publishing, Amsterdam, Netherlands.

Jiang, J.J. and D.W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of ROCLING-97, International Conference on Research in Computational Linguistics*, pages 19–33, Taipei, Taiwan.

Joachims, T. 1997. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 143–151, Nashville, TN, USA, July. Morgan Kaufmann Publisher, San Francisco, CA, USA.

Joachims, T. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 137–142, Chemnitz, Germany, April. Springer-Verlag, London, UK.

Joachims, T. 1999. Making large-scale support vector machine learning practical. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in kernel methods: support vector learning*. MIT Press, Cambridge, MA, USA, pages 169–184.

John, G.H., R. Kohavi, and K. Pfleger. 1994. Irrelevant features and the subset selection problem. In *Proceedings of ICML-94, 11th International Conference on Machine Learning*, pages 121–129, New Brunswick, NJ, USA, July. Morgan Kaufmann Publishers, San Francisco, CA, USA.

Karlgren, J. and M. Sahlgren. 2001. From words to understanding. In Y. Uesaka,

P. Kanerva, and H Asoh, editors, *Foundations of Real-World Intelligence*. CSLI Publications, pages 294–308.

Kehagias, A., V. Petridis, V.G. Kaburlasos, and P. Fragkou. 2003. A comparison of word-and sense-based text categorization using several classification algorithms. *Journal of Intelligent Information Systems*, 21(3):227–247.

Kira, K. and L.A. Rendell. 1992. A practical approach to feature selection. In D. Sleeman and P. Edwards, editors, *Proceedings of ML-92, 9th International Workshop on Machine Learning*, pages 249–256, Aberdeen, UK, July. Morgan Kaufmann Publishers, San Francisco, CA, USA.

Kittler, J. 1978. Feature set search algorithms. In C.H. Chen, editor, *Pattern Recognition and Signal Processing*. Sijthoff & Noordhoff, Alphen aan den Rijin, The Netherlands, pages 41–60.

Kohavi, R. and G.H. John. 1997. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324.

Kohavi, R., P. Langley, and Y. Yun. 1997. The utility of feature weighting in nearest-neighbor algorithms. In W. Gerstner, A. Germond, M. Hasler, and J.D. Nicoud, editors, *Proceedings of ECML-97, 9th European Conference on Machine Learning*, pages 213–220, Prague, Czech Republic, April. Springer.

Koller, D. and M. Sahami. 1996. Toward optimal feature selection. In L. Saitta, editor, *Proceedings of ICML-96, 13th International Conference on Machine Learning*, pages 281–289, Bari, Italy, July. Morgan Kaufmann Publishers, San Francisco, CA, USA.

Kononenko, I. 1994. Estimating attributes: Analysis and extensions of RELIEF. In F. Bergadano and L. de Raedt, editors, *Proceedings of ECML-94, 7th European Conference on Machine Learning*, Lecture Notes in Artificial Intelligence, pages 171–182, Catania, Italy, April. Springer.

Kontostathis, A. 2006. Combining LSI and vector space to improve retrieval performance.

Kontostathis, A. and W.M. Pottenger. 2006. A framework for understanding latent semantic indexing (LSI) performance. *Information Processing and Management*, 42(1):56–73.

Kozima, H. and T. Furugori. 1993. Similarity between words computed by spreading activation on an English dictionary. In *Proceedings of EACL-93, 6th Conference of the European Chapter of ACL*, pages 21–23, Utrecht, Netherlands, April. ACL, Morristown, NJ, USA.

Kraskov, A., H. Stoegbauer, RG Andrzejak, and P. Grassberger. 2005. Hierarchical clustering using mutual information. *Europhysics Letters*, 70(2):278–284.

Kucera, H. and W.N. Francis. 1967. *Computational analysis of present-day American English*. Brown University Press, Providence, RI, USA.

Lam, S.L.Y. and D.L. Lee. 1999. Feature reduction for neural network based text categorization. In *Proceedings of DASFAA-99, 6th IEEE International Conference on Database Advanced Systems for Advanced Application*, pages 195–202, Tapei, Taiwan, April. IEEE Computer Society Press, Los Alamitos, CA, USA.

Lan, M., C.L. Tan, J. Su, and Y. Lu. 2009. Supervised and traditional term weighting methods for automatic text categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):721–735.

Langley, P. 1994. Selection of relevant features in machine learning. In *Proceedings of the AAAI Fall Symposium on Relevance*, pages 1–5, New Orleans, LA, USA. AAAI Press, Menlo Park, CA, USA.

Langley, P. and S. Sage. 1994a. Induction of selective Bayesian classifiers. In R.L. de Mantaras and D. Poole, editors, *Proceedings of UAI-94, 10th Conference*

on Uncertainty in Artificial Intelligence, pages 399–406, Seattle, WA, USA, July. Morgan Kaufmann Publishers, San Francisco, CA, USA.

Langley, P. and S. Sage. 1994b. Oblivious decision trees and abstract cases. In *Working Notes of the AAAI-94 Workshop on Case-Based Reasoning*, pages 113–117, Seattle, WA, USA, July. AAAI Press, Menlo Park, CA, USA.

Langley, P. and S. Sage. 1997. Scaling to domains with irrelevant features. In R. Greiner, editor, *Computational Learning Theory and Natural Learning Systems*, volume 4. MIT Press, Cambridge, MA, USA.

Larkey, L.S. and W.B. Croft. 1996. Combining classifiers in text categorization. In *Proceedings of SIGIR-96, 19th International Conference on Research and Development in Information Retrieval*, pages 289–297, Zürich, Switzerland, August. ACM Press, New York, NY, USA.

Le Cun, Y., J.S. Denker, and S.A. Solla. 1990. Optimal brain damage. *Advances in Neural Information Processing Systems*, 2(1):1990.

Lee, J.H., M.H. Kim, and Y.J. Lee. 1993. Information retrieval based on conceptual distance in is-a hierarchies. *Journal of Documentation*, 49(2):188–207.

Lemarie, PG and Y. Meyer. 1986. Ondelettes et bases hilbertiennes. *Revista Matemática Iberoamericana*, 2(1-2):1–18.

Leopold, E. and J. Kindermann. 2002. Text categorization with support vector machines: How to represent texts in input space? *Machine Learning*, 46(1):423–444.

Lesk, M. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone? In *Proceedings of SIGDOC-86, 5th Annual International Conference on Systems Documentation*, pages 24–26, New York, NY, USA. ACM Press.

Lewis, D.D. 1992. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of SIGIR-92, 15th International*

Conference on Research and Development in Information Retrieval, pages 37–50, Copenhagen, Denmark, June. ACM Press, New York, NY, USA.

Lewis, D.D. 1995. Evaluating and optimizing autonomous text classification systems. In *Proceedings of SIGIR-95, 18th International Conference on Research and Development in Information Retrieval*, pages 246–254, Seattle, WA, USA, July. ACM Press, New York, NY, USA.

Lewis, D.D. 1998. Naive (Bayes) at forty: The independence assumption in information retrieval. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 4–15, Chemnitz, Germany, April. Springer-Verlag, London, UK.

Lewis, D.D. 1999. Reuters-21578 text categorization test collection distribution 1.0.

Lewis, D.D. and M. Ringuette. 1994. A comparison of two learning algorithms for text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 81–93, Las Vegas, NV, USA.

Li, T., Q. Li, S. Zhu, and M. Ogihara. 2002. A survey on wavelet applications in data mining. *SIGKDD Explorations*, 4(2):49–68.

Lin, H.T. and C.J. Lin. 2003. A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods. Technical report, Department of Computer Science, National Taiwan University.

Lin, J. and C. Dyer. 2010. *Data-Intensive Text Processing with MapReduce*. Morgan & Claypool.

Liu, H. and R. Setiono. 1996. A probabilistic approach to feature selection-a filter solution. In L. Saitta, editor, *Proceedings of ICML-96, 13th International Conference on Machine Learning*, pages 319–327, Bari, Italy, July. Morgan Kaufmann Publishers, San Francisco, CA, USA.

Liu, J. and T.S. Chua. 2001. Building semantic perceptron net for topic spotting. In *Proceedings of ACL-01, 39th Annual Meeting on Association for Computational Linguistics*, pages 378–385, Toulouse, France, July. ACL, Morristown, NJ, USA.

Luhn, H.P. 1957. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, 1(4):309–317.

Lyons, J. 1977. *Semantics*. Cambridge University Press, New York, NY, USA.

Mallat, SG. 1989. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693.

Manning, C.D. and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA.

Marill, T. and D. Green. 1963. On the effectiveness of receptors in recognition systems. *IEEE Transactions on Information Theory*, 9(1):11–17.

McHale, M. 1998. A comparison of WordNet and Roget's Taxonomy for measuring semantic similarity. In *Proceedings of COLING-ACL Workshop on Usage of WordNet in Natural Language Processing Systems*, pages 115–120, Montréal, Québec, Canada, August. ACL, Morristown, NJ, USA.

Miller, A.J. 1990. *Subset Selection in Regression*. Chapman and Hall, New York, NY, USA.

Miller, G. and W. Charles. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

Mirsky, L. 1960. Symmetric gage functions and unitarily invariant norms. *The Quarterly Journal of Mathematics*, 11:50–59.

Mitchell, T.M. 1997. *Machine Learning*. McGraw-Hill, New York, NY, USA.

Modrzejewski, M. 1993. Feature selection using rough sets theory. In P. Brazdil, editor, *Proceedings of ECML-93, 6th European Conference on Machine Learning*, pages 213–226, Vienna, Austria, April. Springer.

Mohammad, S. and G. Hirst. 2005. Distributional measures as proxies for semantic relatedness. Submitted for publication.

Moore, A.W. and M.S. Lee. 1994. Efficient algorithms for minimizing cross validation error. In *Proceedings of ICML-94, 11th International Conference on Machine Learning*, pages 190–198, New Brunswick, NJ, USA, July. Morgan Kaufmann Publishers, San Francisco, CA, USA.

Morris, J., C. Beghtol, and G. Hirst. 2003. Term relationships and their contribution to text semantics and information literacy through lexical cohesion. In *Proceedings of CAIS-03, 31st Annual Conference of the Canadian Association for Information Science*, Halifax, Nova Scotia, Canada, May.

Morris, J. and G. Hirst. 1991. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 17(1):21–48.

Moulinier, I., G. Raškinis, and J. Ganascia. 1996. Text categorization: a symbolic approach. In *Proceedings of SDAIR-96, 5th Annual Symposium on Document Analysis and Information Retrieval*, pages 87–99, Las Vegas, NV.

Müller, K.R., S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. 2001. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201.

Nazareth, D.L., E.S. Soofi, and H. Zhao. 2007. Visualizing attribute interdependencies using mutual information, hierarchical clustering, multidimensional scaling, and self-organizing maps. In *Proceedings of HICCS-07, 40th Hawaii International Conference on System Sciences*, volume 40, pages 907–917, Waikoloa, HI, US, January. IEEE.

Ng, H.T., W.B. Goh, and K.L. Low. 1997. Feature selection, perceptron learning,

and a usability case study for text categorization. In *Proceedings of SIGIR-97, 20th International Conference on Research and Development in Information Retrieval*, pages 67–73, Philadelphia, PA, USA, July. ACM Press, New York, NY, USA.

Nigam, K., J. Lafferty, and A. McCallum. 1999. Using maximum entropy for text classification. In *Proceedings of IJCAI-99, 16th International Joint Conference on Artificial Intelligence*, pages 61–67, Stockholm, Sweden, July.

Osgood, C.E. 1952. The nature and measurement of meaning. *Psychological Bulletin*, 49(3):197–237.

Osgood, C.E., G.J. Suci, and P.H. Tannenbaum. 1957. *The Measurement of Meaning.* University of Illinois Press, Urbana-Champaign, IL, USA.

Osuna, E., R. Freund, and F. Girosi. 1997. Training support vector machines: an application to face detection. In *Proceedings of CVPR-97, the IEEE Conference on Computer Vision and Pattern Recognition*, volume 24, Puerto Rico, June. IEEE Computer Society Press, Los Alamitos, CA, USA.

Papoulis, A. 1963. *The Fourier Integral and its Applications.* McGraw-Hill, New York, NY, USA.

Pawlak, Z. 1991. *Rough Sets: Theoretical Aspects of Reasoning about Data.* Kluwer Academic Publishers Norwell, MA, USA.

Peat, H.J. and P. Willett. 1991. The limitations of term co-occurrence data for query expansion in document retrieval systems. *Journal of the American Society for Information Science*, 42(5):378–383.

Peirce, C.S. 1955. Logic as semiotic: The theory of signs. In C.S. Peirce and J. Buchler, editors, *Philosophical Writings of Peirce.* Dover Publications, pages 98–119.

Pfahringer, B. 1995. Compression-based feature subset selection. In *Proceedings of the IJCAI-95 Workshop on Data Engineering for Inductive Learning*, pages

109–119, Montréal, Québec, Canada, August. Morgan Kaufmann Publishers, San Francisco, CA, USA.

Platt, J.C. 1999. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in kernel methods: support vector learning*. MIT Press, Cambridge, MA, USA, pages 185–208.

Polikar, R. 1996. The wavelet tutorial.

Porter, M.F. 1980. An algorithm for suffix stripping. *Program: Electronic Library & Information Systems*, 14(3):130–137.

Rada, R., H. Mili, E. Bicknell, and M. Blettner. 1989. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(1):17–30.

Raghavan, V.V. and S.K.M. Wong. 1986. A critical analysis of vector space model for information retrieval. *Journal of the American Society for Information Science*, 37(5):279–287.

Ratnaparkhi, A. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania.

Resnik, P. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of IJCAI-95, 14th International Joint Conference on Artificial Intelligence*, volume 1, pages 448–453, Montréal, Québec, Canada, August.

Reunanen, J., I. Guyon, and A. Elisseeff. 2003. Overfitting in making comparisons between variable selection methods. *Journal of Machine Learning Research*, 3(7-8):1371–1382.

Rich, E. and K. Knight. 1983. *Artificial Intelligence*. McGraw-Hill, New York, NY, USA.

Richardson, R. and A.F. Smeaton. 1995. Using WordNet in a knowledge-based approach to information retrieval. In *Proceedings of the 17th BCS-IRSG Colloquium on IR Research*, Manchester, UK, April.

Rissanen, J. 1978. Modeling by shortest data description. *Automatica*, 14(5):465–471.

Robertson, S.E. 1990. On term selection for query expansion. *Journal of Documentation*, 46(4):359–364.

Rocchio, J.J. 1971. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, pages 313–323.

Rodriguez, M.D.E.B. and J.M.G. Hidalgo. 1997. Using WordNet to complement training information in text categorization. In *Procedings of RANLP-97, 2nd International Conference on Recent Advances in Natural Language Processing*. John Benjamins Publishing, Amsterdam, Netherlands.

Ron, A. and Z. Shen. 1995. Frames and stable bases for shift-invariant subspaces of $L_2(R^d)$. *Canadian Journal of Mathematics*, 47(5):1051–1094.

Rudin, W. 1987. *Real and complex analysis*. New York.

Ruiz, M.E. and P. Srinivasan. 1999. Hierarchical neural networks for text categorization. In *Proceedings of SIGIR-99, 22nd International Conference on Research and Development in Information Retrieval*, pages 281–282, Berkeley, CA. ACM Press.

Rumelhart, D.E., G.E. Hinton, and R.J. Williams. 1986. *Learning internal representations by error propagation*. MIT Press, Cambridge, MA, USA.

Rumelhart, D.E., B. Widrow, and M.A. Lehr. 1994. The basic ideas in neural networks. *Communications of the ACM*, 37(3):87–92.

Ruskai, M.B., G. Beylkin, R. Coifman, I. Daubechies, S. Mallat, Y. Meyer, and L. Raphael. 1992. *Wavelets and their Applications*. Jones and Bartlett Books in Mathematics, Boston, MA, USA.

Sahlgren, M. 2006. *The Word-Space Model: using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*. Ph.D. thesis, Institutionen för lingvistik. Department of Linguistics, Stockholm University.

Salton, G. and C. Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523.

Salton, G. and M. J. McGill. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, NY, USA.

Salton, G., A. Wong, and C.S. Yang. 1975. A vector space model for information retrieval. *Journal of the American Society for Information Science*, 18(11):613–620.

Salzberg, S. 1991. A nearest hyperrectangle learning method. *Machine Learning*, 6(3):251–276.

Sanderson, M. 2000. Retrieving with good sense. *Information Retrieval*, 2(1):49–69.

Schleif, F.M., M. Lindemann, M. Diaz, P. Maaß, J. Decker, T. Elssner, M. Kuhn, and H. Thiele. 2009. Support vector classification of proteomic profile spectra based on feature extraction with the bi-orthogonal discrete wavelet transform. *Computing and Visualization in Science*, 12(4):1–11.

Schohn, G. and D. Cohn. 2000. Less is more: Active learning with support vector machines. In *Proceedings of ICML-00, 17th International Conference on Machine Learning*, volume 282, pages 285–286, Stanford, CA, USA, June.

Schutze, H., D.A. Hull, and J.O. Pedersen. 1995. A comparison of classifiers and

document representations for the routing problem. *Research and Development in Information Retrieval*, 15:229–237.

Schutze, H. and T. Pedersen. 1997. A co-occurrence-based thesaurus and two applications to information retrieval. *Information Processing and Management*, 3(33):307–318.

Sebastiani, F. 2002. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.

Shawe-Taylor, J. and N. Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA.

Sheikholeslami, G., S. Chatterjee, and A. Zhang. 1998. Wavecluster: A multi-resolution clustering approach for very large spatial databases. In *Proceedings of VLDB-98, 24th International Conference on Very Large Data Bases*, pages 428–439, New York City, NY, USA, August. IEEE.

Singh, M. and G.M. Provan. 1996. Efficient learning of selective Bayesian network classifiers. In L. Saitta, editor, *Proceedings of ICML-96, 13th International Conference on Machine Learning*, pages 453–461, Bari, Italy, July. Morgan Kaufmann Publishers, San Francisco, CA, USA.

Siolas, G. and F. d'Alché Buc. 2000. Support vector machines based on a semantic kernel for text categorization. In *Proceedings of IJCNN-00, IEEE International Joint Conference on Neural Networks*, Austin, TX, USA. IEEE Computer Society Press, Los Alamitos, CA, USA.

Slonim, N. and N. Tishby. 2001. The power of word clusters for text classification. In *Proceedings of ECIR-01, 23rd European Colloquium on Information Retrieval Research*, Darmstadt, Germany.

Smeaton, A.F. and C.J. van Rijsbergen. 1983. The retrieval effects of query expansion on a feedback document retrieval system. *The Computer Journal*, 26(3):239–246.

Smola, A.J., B. Schölkopf, and K.R. Müller. 1998. The connection between regularization operators and support vector kernels. *Neural Networks*, 11(4):637–649.

Steinbach, M., G. Karypis, and V. Kumar. 2000. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*.

Stoppiglia, H., G. Dreyfus, R. Dubois, Y. Oussar, I. Guyon, and A. Elisseeff. 2003. Ranking a random feature for variable and feature selection. *Journal of Machine Learning Research*, 3(7-8):1399–1414.

Sussna, M. 1993. Word sense disambiguation for free-text indexing using a massive semantic network. In *Proceedings of CIKM-93, 2nd International Conference on Information and Knowledge Management*, pages 67–74, Washington, DC, USA, November. ACM Press, New York, NY, USA.

Szu, H.H., B.A. Telfer, and S.L. Kadambe. 1992. Neural network adaptive wavelets for signal representation and classification. *Optical Engineering*, 31:1907.

Tishby, N., F.C. Pereira, and W. Bialek. 2000. The information bottleneck method. *Arxiv preprint physics/0004057*.

Tong, S., D. Koller, and L.P. Kaelbling. 2001. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2(1):45–66.

Tuntisak, S. and S. Premrudeepreechacharn. 2007. Harmonic detection in distribution systems using wavelet transform and support vector machine. In *Proceedings of Powertech-07, Conference of the IEEE Power Engineering Society*, pages 1540–1545, Lausanne, Switzerland, July.

Turney, P.D. 2001. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of ECML-01, 12th European Conference on Machine Learning*, pages 491–502, Freiburg, Germany, September.

Unser, M. 1997. Ten good reasons for using spline wavelets. In *Proceedings of SPIE, Wavelet Applications in Signal and Image Processing V*, volume 3169, pages 422–431.

Unser, M. and A. Aldroubi. 1993. Polynomial splines and wavelets: A signal processing perspective. In *Academic Press Wavelet Analysis And Its Applications Series*. Academic Press Professional, Inc. San Diego, CA, USA, pages 91–122.

Ureña López, L.A., M. Buenaga, and J.M. Gómez. 2001. Integrating linguistic resources in text classification through WSD. *Computers and the Humanities*, 35(2):215–230.

Uschold, M. and M. Gruninger. 1996. Ontologies: Principles, methods and applications. *Knowledge Engineering Review*, 11(2):93–136.

van Rijsbergen, C. J. 1979. *Information Retrieval*. Butterworths, London, UK.

van Rijsbergen, C. J. 2004. *The Geometry of Information Retrieval*. Cambridge University Press, New York, NY, USA.

Vapnik, V.N. 1998. *Statistical learning theory*. John Wiley & Sons, New York, NY, USA.

von Uexküll, J. 1982. The theory of meaning. *Semiotica*, 42(1):25–82.

Voorhees, E.M. 1994. Query expansion using lexical-semantic relations. In *Proceedings of SIGIR-94, 17th International Conference on Research and Development in Information Retrieval*, Dublin, Ireland. ACM Press, New York, NY, USA.

Weaver, H.J. 1988. *Theory of Discrete and Continuous Fourier Analysis*. John Wiley & Sons, New York, NY, USA.

Weigend, A.S., E.D. Wiener, and J.O. Pedersen. 1999. Exploiting hierarchy in text categorization. *Information Retrieval*, 1(3):193–216.

Weston, J., A. Elisseeff, B. Scholkopf, M. Tipping, and L.P. Kaelbling. 2003. Use of the zero-norm with linear models and kernel methods. *Journal of Machine Learning Research*, 3(7-8):1439–1461.

Weston, J., S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. 2000. Feature selection for SVMs. *Advances in Neural Information Processing Systems*, 13:668–674.

Wettschereck, D. and D.W. Aha. 1995. Weighting features. In *Proceedings of ICBR-95, 1st International Conference on Case-Based Reasoning*, pages 347–358, Sesimbra, Portugal, October. Springer.

Wiener, E., J.O. Pedersen, and A.S. Weigend. 1995. A neural network approach to topic spotting. In *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 317–332, Las Vegas, NV.

Wilks, Y., D. Fass, C. Guo, J.E. McDonald, T. Plate, and B.M. Slator. 1990. Providing machine tractable dictionary tools. *Machine Translation*, 5(2):99–154.

Wittek, P. 2006. Using pseudo-relevance feedback to filter search results. Technical report, A*Star Institute for Infocomm Research.

Wittek, P. 2007. Information retrieval by continuous functions. Master's thesis, Budapest University of Technology and Economics.

Wittek, P. and S. Darányi. 2007. Representing word semantics for IR by continuous functions. In S. Dominich and F. Kiss, editors, *Studies in Theory of Information Retrieval. Proceedings of ICTIR-07, 1st International Conference of the Theory of Information Retrieval*, pages 149–155, Budapest, Hungary, October. Foundation for Information Society.

Wittgenstein, L. 1967. *Philosophical Investigations*. Blackwell Publishing, Oxford, UK.

Wong, S.K.M. and V.V. Raghavan. 1984. Vector space model of information retrieval: A re-evaluation. In *Proceedings of SIGIR-84, 7th International Conference on Research and Development in Information Retrieval*, pages 167–185, Cambridge, England. ACM Press, New York, NY, USA.

Wong, S.K.M., W. Ziarko, and P.C.N. Wong. 1985. Generalized vector space model in information retrieval. In *Proceedings of SIGIR-85, 8th International Conference on Research and Development in Information Retrieval*, pages 18–25, Montréal, Québec, Canada. ACM Press, New York, NY, USA.

Yang, Y. 1999. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1):69–90.

Yang, Y. and C.G. Chute. 1994. An example-based mapping method for text categorization and retrieval. *ACM Transactions on Information Systems*, 12(3):252–277.

Yang, Y. and X. Liu. 1999. A re-examination of text categorization methods. In *Proceedings of SIGIR-99, 22nd International Conference on Research and Development in Information Retrieval*, pages 42–49, Berkeley, CA, USA, August. ACM Press, New York, NY, USA.

Yang, Y. and J.O. Pedersen. 1997. A comparative study on feature selection in text categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning*, volume 97, pages 412–420, Nashville, TN, USA, July. Morgan Kaufmann Publishers, San Francisco, CA, USA.

Yu, H., J. Yang, and J. Han. 2003. Classifying large data sets using SVMs with hierarchical clusters. In *Proceedings of SIGKDD-03, 9th International Conference on Knowledge Discovery and Data Mining*, pages 306–315, Washington, DC, USA, August. ACM Press, New York, NY, USA.

Zhang, L., W. Zhou, and L. Jiao. 2004. Wavelet support vector machine. *IEEE Transactions on Systems, Man, and Cybernetics*, 34(1):34–39.

Zhang, T., R. Ramakrishnan, and M. Livny. 1996. BIRCH: an efficient data clustering method for very large databases. In *Proceedings of SIGMOD-96, International Conference on Management of Data*, pages 103–114, Montréal, Québec, Canada, June. ACM Press, New York, NY, USA.

Zipf, G.K. 1935. *The psychobiology of language.* Houghton Mifflin, Boston, MA, USA.

Zipf, G.K. 1949. *Human behavior and the principle of least effort: An introduction to human ecology.* Addison-Wesley, Harlow, UK.